MSc. Project Report

A Notation Selection Tool For MathML

by
Dicheng Liu

Supervisors: Dr. Stephen Watt
Dr. Rob Corless

DEPARTMENT OF COMPUTER SCIENCE

Submitted in partial fulfillment of the requirement
for the degree of Master of Computer Science

FACULTY OF GRADUATE STUDIES
THE UNIVERSITY OF WESTERN ONTARIO
LONDON, ONTARIO

January, 2001

Abstract

For a mathematical object, there usually exist a few notations to represent it. Consequently, for a given mathematical idea (object), there is a certain set of Content MathML markups to encode it, but there might be a few different sets of Presentation MathML markups to represent it each of which corresponds to one of its notations. People from different background have different mathematical notational conventions. In this project, we have developed an extensible notation selection tool which provides a graphical user interface (GUI) to allow the user to select his/her notational conventions while transforming content MathML to presentation MathML.

Acknowledgements

I would like to express my most sincere gratitude to my supervisors, Dr. Stephen Watt and Dr. Robert Corless, for their constant encouragement, invaluable guidance, as well as generous support.

I would also like to thank all other people in the Department of Computer Science, the University of Western Ontario who has contributed to the completion of my project and graduate studies, especially the friends of the Symbolic Computation Lab.

Special thanks to my family, especially to my husband, Manqing, for his love, understanding, and support.

Contents

1.Introduction

Mathematical ideas exist independently of the notations that represent them. People from different backgrounds (for example, different countries or areas of mathematics) have different notational conventions. In other words, for a mathematical object, there usually exist a few notations to represent it. For example, the mathematical idea "A times B" can be represented by "$A \cdot B$", "$A \times B$", or "AB".

MathML[1],[2] is an XML-based[3] standard produced by the World Wide Web Consortium (W3C). There are two ways of encoding mathematical data using MathML: Content MathML markup and presentation MathML markup. Content markup is concerned with the semantics of mathematics. Presentation markup is concerned with the rendering of mathematics. Therefore, a given mathematical idea will use a certain set content MathML markup to encode it, but there might be several different choices of presentation MathML markup to represent the different notations.

Presentation markup is more general than Content markup in the sense that MathML has presentation tags for almost any mathematical notation but content tags for only a fixed set of mathematical concepts. It should be recalled that an XSLT processor (such as XT or SAXON) enables the rendering of Content markup by first transforming content markups to presentation markups[4]. XSLT[5] is a language designed by *W3C* for transforming XML documents into other XML documents.

We have designed and implemented a notation selection tool which provides a graphical user interface (GUI) to allow a user to select his/her notational conventions while transforming content MathML to presentation MathML. This is a Java application which produces a XSLT stylesheet.

We have selected a realistic set of mathematical notations that this tool can provide to the user. However, it is not possible to foresee all the notations which different group

may want to use. With the development of mathematical research, new notations might be created. Therefore, the notation selection tools should be extensible, i. e. it is necessary for these tools to provide the user with new notations easily. To achieve the tool's extensibility, we use an XML document as an input file to organize mathematical notations so that we only need to add some elements to the XML document instead of modifying the Java program in order to add more mathematical items or notations to the application.

In this report, we describe the background of our work, the selected notations for different choices, and the design and implementation details of the notation selection tool for the transformation of Content MathML to Presentation MathML. The reminder of this report is organized as follows: Section 2 presents the different mathematical notations offered by this application. In section 3, we introduce some software related background , including some concepts related to XML, MathML, XSLT and graphical user interfaces (GUIs) and the Java Swing library. Section 4 is a detailed description of the tool we have developed, including the structure of the input XML document, and the design and implementation of the application. We chose Java to implement this tool, as Java is becoming a popular and well-developed programming language in the world of GUI development. The whole work is summarized in section 5. The input XML document is given in Appendix A. Appendix B is Java source code for this tool.

2. Mathematical Notations

We have selected a realistic initial set of notations for mathematical items. These items are grouped into different catalogs, such as ARITHMETIC, CALCULUS, COMBINATIONS, CONTINUED FRACTIONS, INTERVALS, LINEAR ALGEBRA, LOGARITHM, POWER OR ROOT, and TRIGONOMETRY. Each catalog consists of a number of mathematical items, and each item consists of a list of choices. For example, the ARITHMETIC catalog consists of DIVISION, and MULTIPLICATION items. An item consists of two or more notational choices. For example, the MULTIPLICATION item has three different notational choices, i.e. "A multiplied by B" can be presented by "A · B", "A × B", or "AB".

So far this notation selection tool provides the user with the notations described in this section. This tool is extensible, i.e. we can easily add more notations to this tool for the user to select as needed later. To add some notations, we need only add some corresponding elements to the input XML document.

2.1 Arithmetic

Division: the idea "a divided by b" can be represented as

$a \div b$, a/b, $\frac{a}{b}$, or $b\overline{)a}$

Multiplication: "a times b" might be rendered as

$a \times b$, $a \cdot b$, or ab

Continued Fractions: The continued fraction with partial quotient $a_0, a_1, a_2, \ldots$ can be represented as

$a_0 + \frac{1}{a_1 + \ldots}$, $[a_0, a_1, \ldots]$, or $a_0 + \frac{1|}{|a_1} + \ldots$

2.2 Calculus

Derivative: The derivative of f with respect to x can be represented as

$\frac{df}{dx}$, $D_x f$, or f_x

Partial Derivative: The partial derivative of f with respect to x can be represented as

$\frac{\partial f}{\partial x}$, $\partial_x f$, $D_x f$, $\bigtriangledown_x f$, or f_x

2.3 Binomial Coefficients

The number of ways of choosing m objects from a collection of n distinct objects without regard to order can be represented as one of the following notations:

$\binom{n}{m}$ C_m^n C_n^m ${}_nC^m$

2.4 Intervals:

Open interval can be represented as (a, b), or $]a, b[$.

Consequently, the **open-closed interval** would be written as (a, b], or]a, b],

And the **closed-open interval** would be written as [a, b), or [a, b[

2.5 Power or Root

Exponent: The power *x* of *e*(the base of the natural logarithms)

might be written as $\exp(x)$, or e^x

Root: the square root can be written as $\sqrt{a}$, or $a^{1/2}$.

Similarly, the n-th root can be written as $\sqrt[n]{a}$ or $a^{1/n}$

2.6 Logarithms

The logarithm of x with base 2 can be written as lg(x), $\log_2(x)$, or log(x)

The logarithm of x with base 10 can be written as log(10), or $\log_{10}(x)$

The logarithm of x with base *e* can be written as *ln*(x) , or log(x)

2.7 Linear Algebra

Vector: The name of a vector can be written as

$\mathbf{V}$, V, $\vec{V}$, or V_i

Matrix: A matrix name might be written as

$\mathbf{M}$, M, $\overleftrightarrow{M}$, M_{ij}, or $[M_{ij}]$

2.8 Trigonometry

Tangent : tangent of x can be represented as *tan(x)* , or *tg(x)*.

Cotangent: cotangent of x can be written as *cotan(x)*, *ctg(x)*, or *cot(x)*.

Arcsine: arcsine of x can be written as $sin^{-1}(x)$, *arcsin(x)*, or *asin(x)*.

All these notations are displayed as images on the GUI of this tool for the user to select.

3. Background

3.1 XML

XML (eXtensible Markup Language) is "a data format for storing structured and semi-structured text intended for communication or publication in a variety of media. An XML document contains special instructions, called tags, which enclose identifiable parts of the document"[6].

Here is a sample fragment of an XML document:

```
<EMPLOYEE>
      <NAME>
            <FIRSTNAME> Tina</FIRSTNAME>
            <LASTNAME> Li</LASTNAME>
      </NAME>
      <SIN> 123 456 789</SIN>
      <SALARY> 60000</SALARY>
      <DATE-HIRED>
            <YEAR> 1999</YEAR>
            <MONTH> 01</MONTH>
            <DAY> 01</DAY>
      </DATA-HIRED>
</EMPLOYEE>
```

XML is a simplified subset of SGML(Standard General Markup Language). It inherits SGML's three main advantages: Generalized and descriptive; extensible; and validated. The XML format is similar to HTML(Hypertext Markup Language). However, HTML has a fixed set of tags, whereas XML allows user to define tags. This provides individuals with the ability to describe the data with meaningful tags. Its extensibility makes XML suitable for describing complex data structure.

There exist many XML parsers for parsing and validating XML documents. In our work, we use the XERCES_1-3-1 parser, which was developed in Java by APACHE to parse our input XML document in order to extract data from the XML document.

3.2 MathML

MathML (Mathematical Markup Language) is an application of XML. "MathML is a format for describing mathematics as a basis for machine-to-machine communication"[6]. "It is intended to facilitate the use and reuse of mathematical and scientific content on the web and for other applications such as computer algebra systems, and print typesetters. MathML can be used to encode both mathematical notations for high quality visual display, and mathematical content for semantic applications such as scientific software"[6].

There are three categories of MathML elements: presentation elements, content elements, and interface elements.

MathML presentation elements describe mathematical notation's visually oriented two-dimensional structure. They are concerned with layout and rendering of the mathematical notations.

For example, "*a* times *b*" can be rendered as :
$a \times b,$ by the following Presentation markups:

```
<mrow>
    <mi>a</mi>
    <mo>&times;</mo>
    <mi>b</mi>
<mrow/>
```

a • *b* by

```
<mrow>
    <mi>a</mi>
    <mo>&middot;</mo>
    <mi>b</mi>
<mrow/>
```

or *ab* by

```
<mrow>
      <mi>a</mi>
      <mo>&InvisibleTimes;</mo>
      <mi>b</mi>
<mrow/>
```

An intermediate working draft of MathML 2.0 introduced the <mchar> tag, and therefore appears in this project. However, in the end, <mchar> was not adopted and entities must be used in their places, e.g. `<mo>&InvisibleTimes;</mo>` instead of `<mo><mchar name = "InvisibleTimes"/></mo>`

MathML content elements describe mathematical objects directly. They encode the mathematical meaning of an expression.

e.g "a times b" can be encoded as

```
<apply>
     <times/>
     <ci>a</ci>
     <ci>b</ci>
</apply>
```

MathML interface elements provide a mechanism for embedding MathML expressions within an HTML page, and for passing the necessary information between an HTML browser and a MathML helper application doing the MathML processing.

3.3XSL/XSLT

XSL (eXtensible Stylesheet Language) is a language for expressing style sheets for XML. To present an XML document on paper or screen, the use of style sheets is essential. The content of an XML element has no explicit style. XSL must specify how each element should be presented according to what the element is and how it occurs.

There are two steps for the presentation process. First, the result tree, which is created from the XSL processor, is constructed from the source tree, which is parsed by an XML parser from the source XML document. Second, the result tree is interpreted to produce the formatted output on a display, on paper, or other medium.

XSLT (XSL Transformations) is a language for transforming XML documents into other XML documents based on a set of pattern matching rules.

Our tool will, according to the user's selections, create a style sheet in XSLT for the transformation from content MathML to presentation MathML. It uses some existing tools (processors), e.g. XT, or SAXON, to perform the transformations, i.e. to create an output presentation MathML document from the input content MathML document. Using the stylesheet our tool creates, the tool then uses some mathematical viewer , such as WebEQ, to display the mathematical object.

3.4 Java Swing and Graphical User Interfaces

Graphical user interfaces (GUIs) have become quite popular in today's software applications. With the advances in operating systems, such as Unix Motif or Windows, and software programming languages, such as Java, graphical user interfaces (GUIs) are enjoying rich growth and several tools have been created to write GUIs.

`Swing` is a Java GUI library. It is part of the Java Foundation Classes (JFC). `Swing` contains a rich and convenient set of user interface components that is much more complete, flexible, and portable than Java's AWT (Abstract Window Toolkit). It provides many attractive features for writing large-scale applications. First, it does not rely on the underlying runtime platform because it is written in "pure Java" and therefore provides good portability. Second, the "Pluggable Look-and-Feel" feature: `Swing` is capable of dictating the "*look-and-feel"* of each of the components, even resetting the look-an-feel at runtime. The programmer can choose among several pre-built "*look-and-feels*", or can create his/her own "look-and-feel". Third, `Swing` components were designed with the "Model-View-Control" design pattern. This makes a very clear distinction between the data(the "model") and the actual display(the "view") of a component. This separation means that components are very flexible. Our tool is written in Java and is based on the Swing library.

4. A Notation Selection Tool For Content-to-Presentation Transformation of MathML

4.1 Overview

There already exist some style sheets for the content-to-presentation MathML transformation. However, these style sheets provide the user with only one presentation for each content markup. In other words, there is only one notation for each mathematical object. The user can not select his/her notational conventions. It is desirable to develop notation selection tools to allow a user to select his/her preferred notations while transforming content MathML into presentation MathML.

We have developed a notation selection tool, which provides the user with a graphic user interface to allow the user to select his/her preferred notations, for the transformation from content MathML to presentation MathML. For each mathematical notation, there is a set of XSLT templates, which will override the corresponding default templates in the input style sheet for the content-to-presentation transformation.

4.2 The Input XML Document

In our work, we use an XML document as input to organize the notations, so that the tool will be extensible, i.e. we can easily add more mathematical catalogs, items, or notations for the user to select as needed later. To achieve this goal, we need only add some elements to this XML document.

Here is the structure of our input XML document:

```
<?xml version="1.0"?>
<mnotations>
    <catalog>  <!-- ARITHMETIC   -->
        <name>Arithmetic</name>
        <itemlist>
           <item>
              <keyword> DIVISION </keyword>
              <choicelist>
                 choice>
                    <image src = "div1.gif"/>
                    <keyvalue> 1 </keyvalue>
                    <presentation>
                      <!--  templates to this notation  -->
                    </presentation>
                 </choice>
                 .
                 .     <!-- other choices of DIVISION -->
                 .
              </choicelist>
           </item>
           .
           .      <!—other Items of ARITHMETIC -->
           .
        </itemlist>
    </catalog>
   .
   .   <!--  other CATALOGS   -->
</mnotations>
```

In this XML document, we use a CATALOG element for each group of mathematical items, for example, the ARITHMETIC catalog. Each CATALOG element has one or more child-elements named ITEM elements. Each ITEM element corresponds to a mathematical item, such as DIVISION item. Inside an ITEM element, are the TITLE element, which contains the title of the item, KEYWORD element, which contains the keyword of this item, and CHOICELIST element, which contains a list of choices for this item. There are two or more CHOICE elements as child elements of CHOICELIST.

Each CHOICE element has the following child elements: a KEYVALUE element containing the key value used for identifying this choice; a IMAGE element specifying the visualization of this choice; a PRESENTATION element containing the XSLT templates corresponding to this choice for the transformation from content markup to presentation markups.

We use an input XSLT stylesheet which contains the default templates for the transformation. When a user selects a notation for an ITEM, the templates inside the corresponding PRESENTATION will be appended to the input style sheet. These templates will override the default templates provided in the input style sheet when performing the transformation because of their higher priority.

Appendix A is our input XML document. All the notations are organized into mathematical groups. For example, we have selected two mathematical items (DIVISION and MULTIPLICATION) which realistically have multiple notations.

4.3 The Structure of This Tool

This tool is implemented in Java because of Java's good portability and its rich and convenient user interface package, the SWING library.

To develop this tool, we first need a XML parser to parse our input XML document. There exist many XML parsers for us to use. We selected the *xerces-1_1_3* package[9], which contains an XML parser, and is implemented in Java. It provides a stable working environment and convenient documentation API (Application Programming Interface).

We model the application as shown in the OMT diagram in *figure 4.1*:

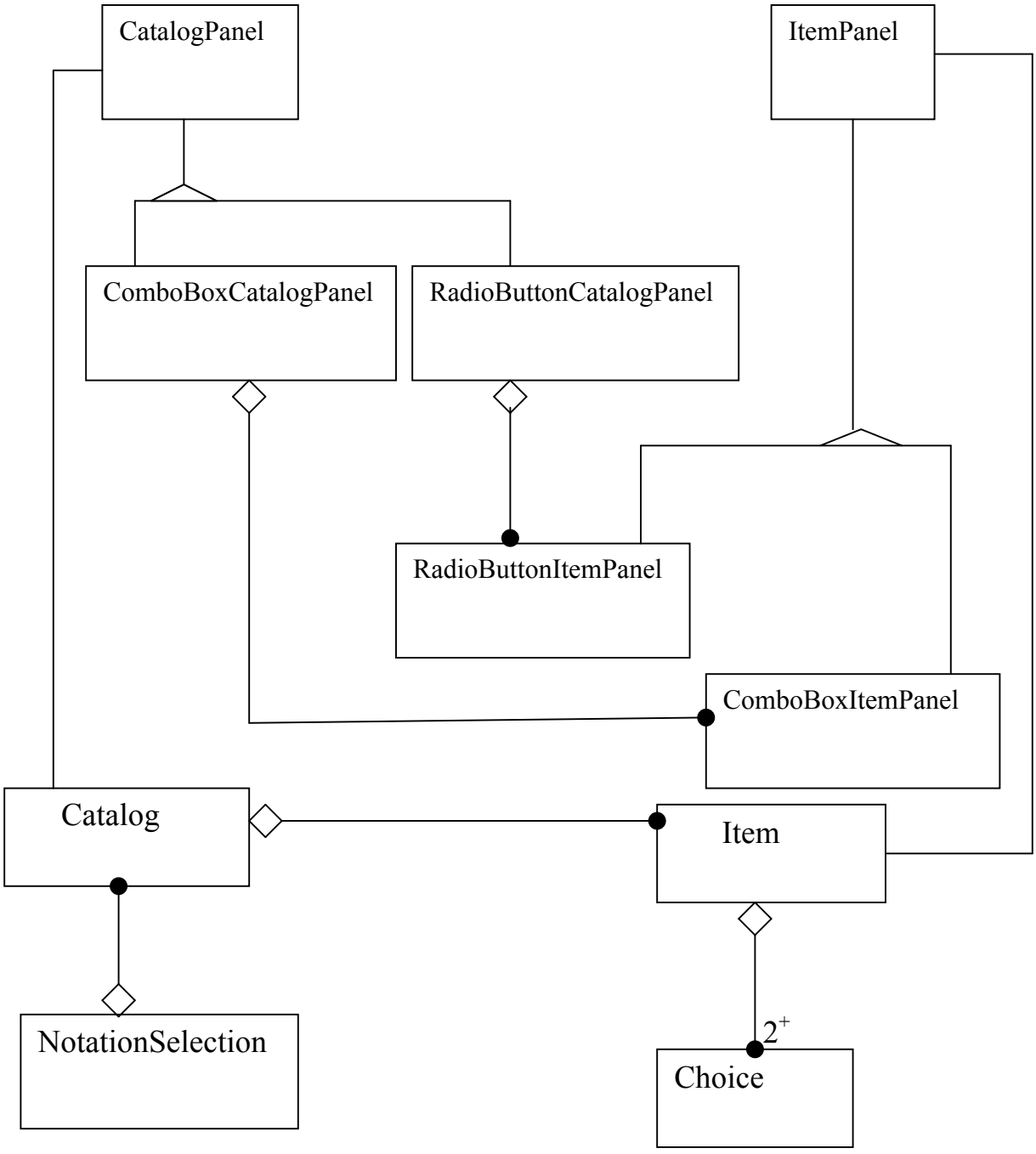

Figure 4-1: OMT diagram

At first, the system parses the input XML document. Then it extracts the data from the parsed tree and creates a linked list of `Catalog` objects, and each `Catalog` object contains a list of `Items`, each of which has a list of `Choices`. In the GUI of this tool, a `CatalogPanel` object represents each Catalog object, which is either a `ComboBoxCatalogPanel` object or `RadioButtonCatalogPanel` object. The `CatalogPanel` is a subclass of the `JPanel` class, which is a `JComponent` in the Java `Swing` package. An `ItemPanel` object represents each `Item` object, which is either a `ComboBoxItemPanel` object or A `RadioButtonItemPanel`. The `ItemPanel` is also a subclass of the `JPanel` class. A `ComboBoxCatalogPanel` object contains a number of `ComboBoxItemPanel` objects in which its `Choices` are grouped as a `JComboBox` list. Similarly, a `RadioButtonCatalogPanel` object contains a number of `RadioButtonItemPanel` objects in which its `Choices` are presented as a group of `JradioButton` objects.

4.4 Using The Tool

This tool can provide the user with two different styles of graphical user interfaces, i. e. DropDown (ComboBox) style and RadioButton style, which are shown in Figure 4-2, and Figure 4-3 respectively. The user can choose his/her preferred GUI. Basically, the Dropdown style GUI can provide the user with more notational choices but it is not so intuitive and convenient to use. The user has to click on every drop-down box to see all the notations. In contrast, the RadioButton style GUI is very convenient for the user to select his/her preferred notations, but the number of notations for an ITEM are limited because of the limitation of the width of the GUI.

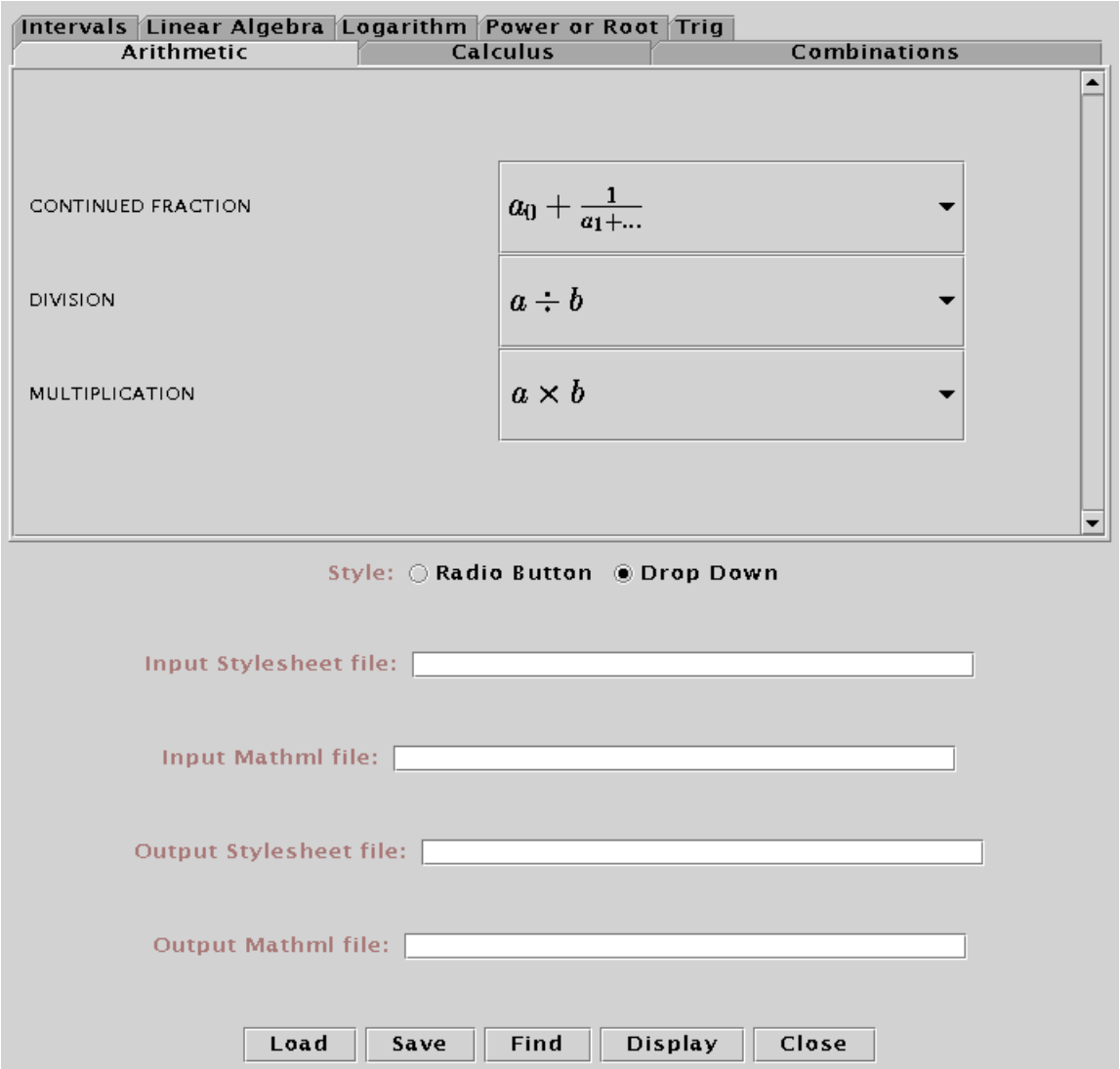

Figure 4-2: DropDown (ComboBox) Style Graphical User Interface

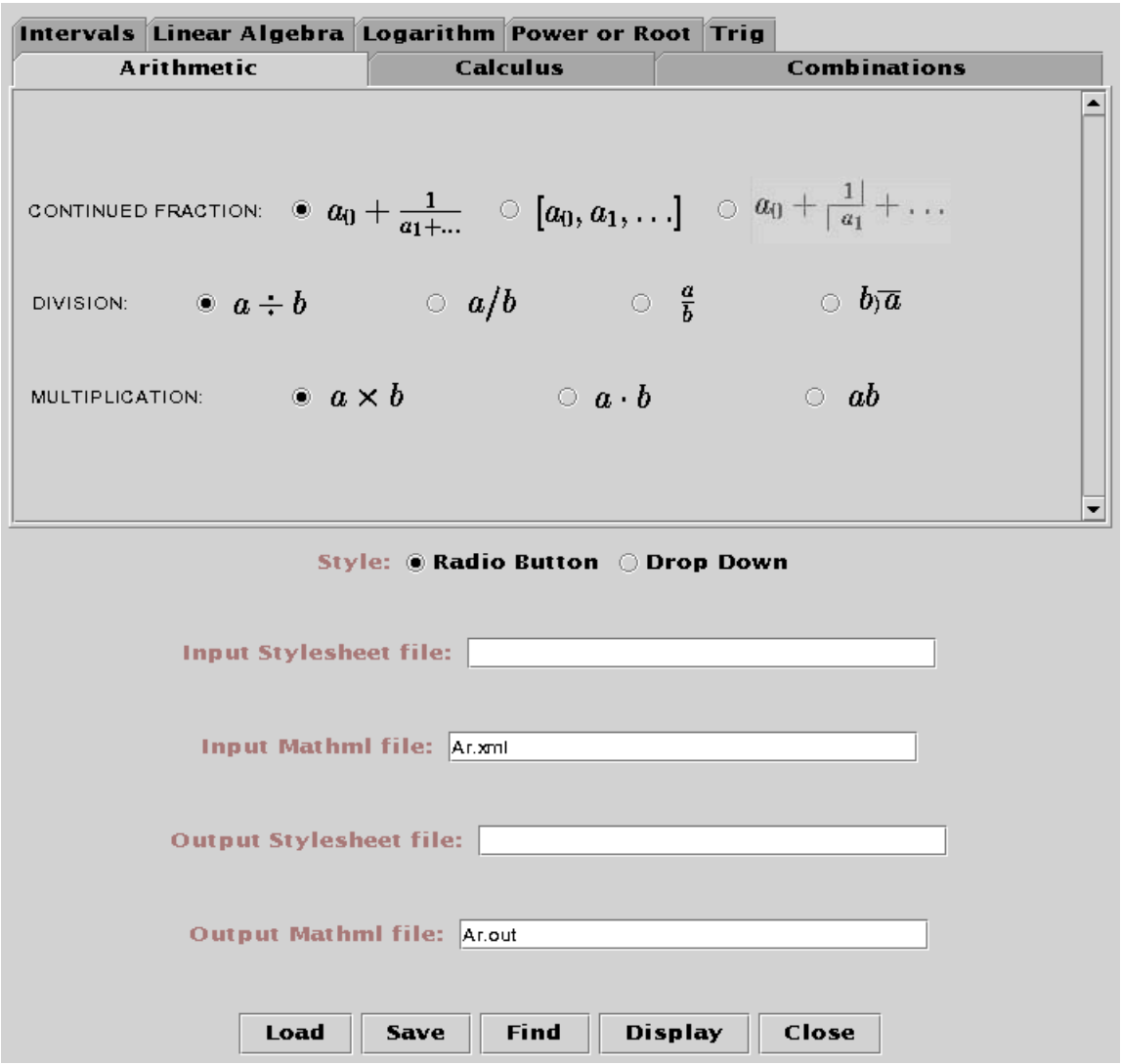

Figure 4-3 RadioButton Style Graphical User Interface

After the user has selected his/her preferred notations, he/she can save his/her selections by clicking on the *Save* button on the GUI. If the user does not select a notation for an Item, the first notation (choice) is saved as the default choice for this Item. While saving the choices, the user might save either the key value for identifying the choices or the style sheet for the transformations, or both the key values and the style sheet, as shown in figure 4-4.

Figure 4-4: the Save window

When a user needs to modify his/her previous selections after exiting the tool, he/she can load his/her previous saved selections by clicking on the *Load* button and do some desired reselections. He/she does not need to start over and reselect for all items again.

The user might provide an input content MathML document(the default is show.xml) and an input XSLT style sheet(default is mml2mml.xsl[4]), and then specify the output style sheet name(default is mml2mml.new) and the file name of transformed presentation MathML document(default is show.out). The resulting presentation MathML document can be displayed on the screen by clicking on the *Display* button on the GUI. The underlying work for *Display* includes letting the XSLT processor XT perform the content-to-presentation transformation using the output style sheet and calling the Mathematical viewer named *Viewer* to render the transformed presentation MathML document on the screen.

Here are some examples showing the transformations done by this tool. The tool took an input Content MathML document named Ar.xml shown as follows:

```
<?xml version="1.0"?>

<math xmlns="http://www.w3.org/1998/Math/MathML"
xmlns:mmlx="www.orcca.on.ca/MathML/Extension0">

<!--    ** TEST ARITHMETIC **  -->

<apply>
  <mmlx:contfrac/>
  <cn>3</cn>
  <cn>7</cn>
  <cn>1</cn>
  <cn>292</cn>
  <cn>1</cn>
  <cn>1</cn>
  <cn>1</cn>
  <cn>2</cn>
  <cn>1</cn>
</apply>

<apply>
    <divide/>
    <ci>a</ci>
    <cn>5</cn>
</apply>

  <apply>
    <times/>
    <ci>a</ci>
    <cn>5</cn>
  </apply>

</math>
```

When we selected the notations shown in Figure 4-3, and then clicked on the *Display* button, the tool created a new stylesheet named mml2mml.new(the default name) with which it transformed the input Content MathML document Ar.xml to the output Presentation MathML Ar.out shown as :

```
<?xml version="1.0" encoding="utf-8"?>
<mrow xmlns:mml="http://www.w3.org/1998/Math/MathML"
xmlns:mmlx="www.orcca.on.ca/MathML/Extension0"
xmlns="http://www.w3.org/1998/Math/MathML">

<mrow>
```

```
<mn>3</mn>
<mo>+</mo>
<mfrac>
<mn>1</mn>
<mrow>
<mn>7</mn>
<mo>+</mo>
<mfrac>
<mn>1</mn>
<mrow>
<mn>1</mn>
<mo>+</mo>
<mfrac>
<mn>1</mn>
<mrow>
<mn>292</mn>
<mo>+</mo>
<mfrac>
<mn>1</mn>
<mrow>
<mn>1</mn>
<mo>+</mo>
<mfrac>
<mn>1</mn>
<mrow>
<mn>1</mn>
<mo>+</mo>
<mfrac>
<mn>1</mn>
<mrow>
<mn>1</mn>
<mo>+</mo>
<mfrac>
<mn>1</mn>
<mrow>
<mn>2</mn>
<mo>+</mo>
<mfrac>
<mn>1</mn>
<mrow>
<mn>1</mn>
<mo>+</mo>
<mo>...</mo>
</mrow>
</mfrac>
</mrow>
</mfrac>
</mrow>
</mfrac>
</mrow>
</mfrac>
</mrow>
</mfrac>
</mrow>
</mfrac>
```

```
</mrow>
</mfrac>
</mrow>


<mrow>
<mi>a</mi>
<mo>
<mchar name="divide"/>
</mo>
<mn>5</mn>
</mrow>


<mrow>
<mi>a</mi>
<mo>
<mchar name="times"/>
</mo>
<mn>5</mn>
</mrow>

</mrow>
```

The above output Presentation MathML document was displayed as in Figure 4-5

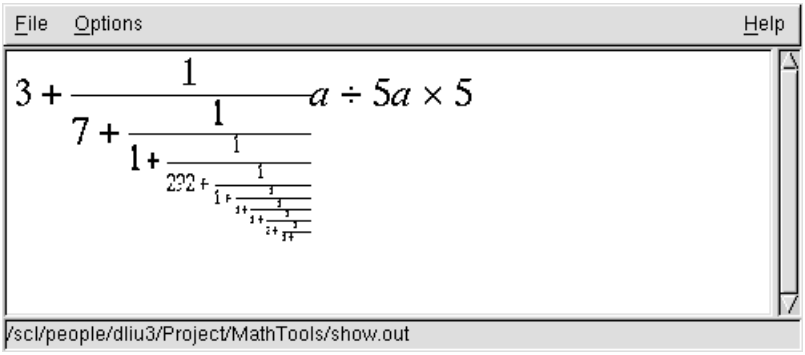

Figure 4-5: The Displayed Output Corresponding to the Choices shown in Figure 4-3

When we selected the notations shown as in Figure 4-6 and clicked on the *Display* button again, the tool created another output stylesheet, the output Presentation MathML

document was as follows, and this output Presentation MathML document was displayed as in Figure 4-7.

```
<?xml version="1.0" encoding="utf-8"?>
<mrow xmlns:mml="http://www.w3.org/1998/Math/MathML"
xmlns:mmlx="www.orcca.on.ca/MathML/Extension0"
xmlns="http://www.w3.org/1998/Math/MathML">

<mrow>
<mfenced close="]" open="[">
<mn>3</mn>
<mn>7</mn>
<mn>1</mn>
<mn>292</mn>
<mn>1</mn>
<mn>1</mn>
<mn>1</mn>
<mn>2</mn>
<mn>1</mn>
<mi>...</mi>
</mfenced>
</mrow>


<mrow>
<mi>a</mi>
<mo>/</mo>
<mn>5</mn>
</mrow>


<mrow>
<mi>a</mi>
<mo>
<mchar name="middot"/>
</mo>
<mn>5</mn>
</mrow>


</mrow>
```

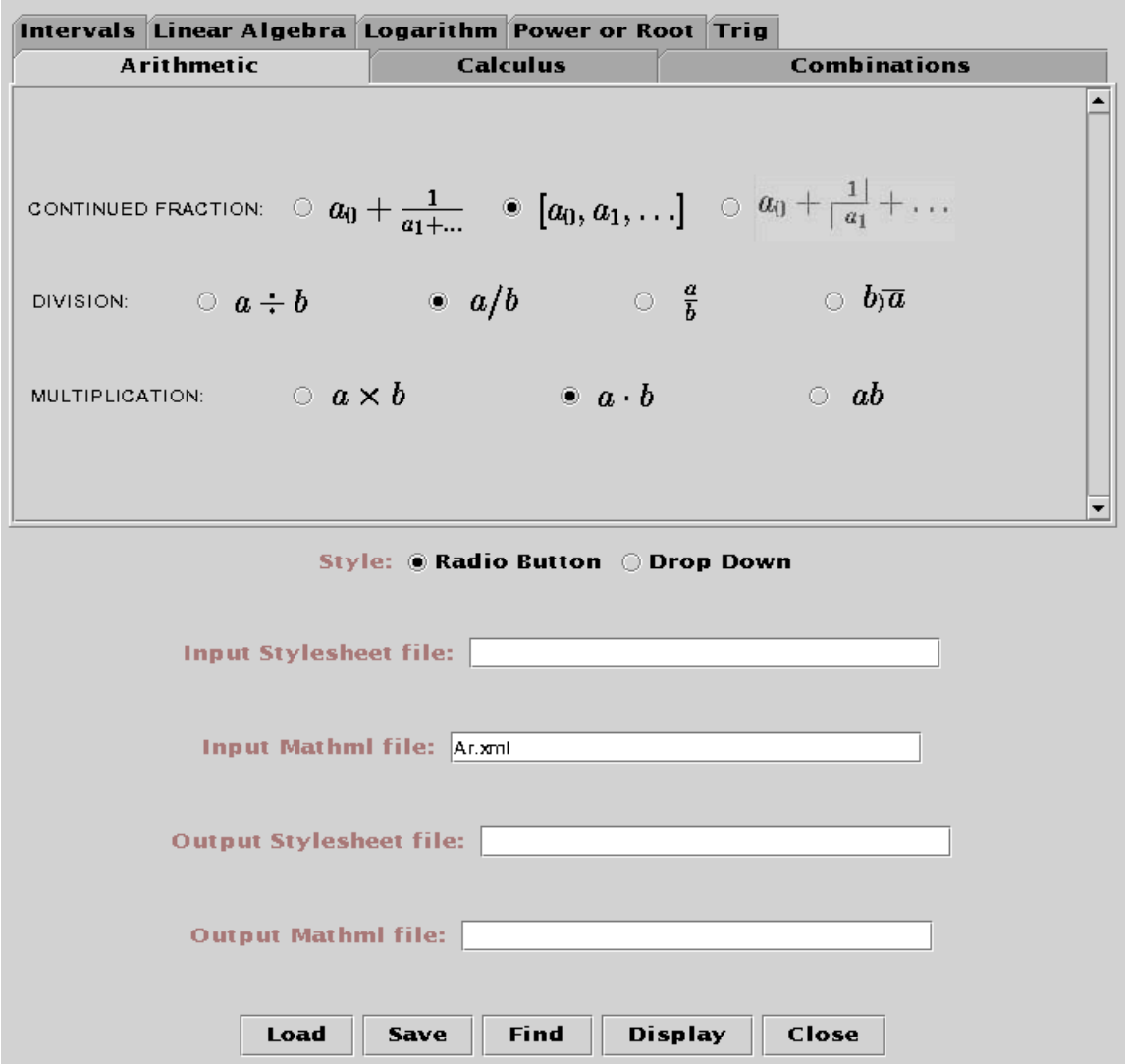

Figure 4-6: Select the Second Notations

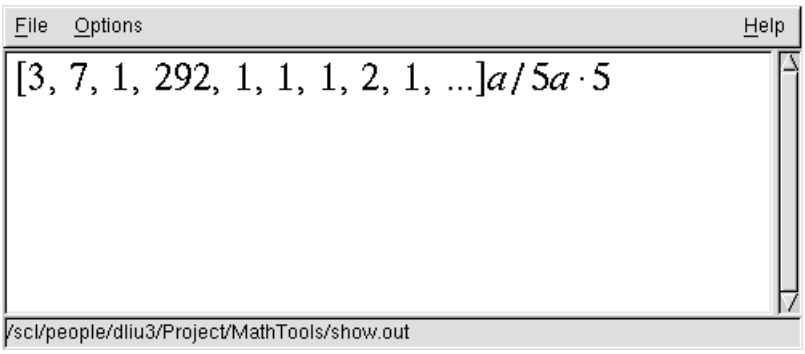

Figure 4-7: The Displayed Output Corresponding to the Choices shown in Figure 4-6

To find some item by keyword, one needs to click on the "Find" button. A *Find* window will pops up as shown in figure 4-5. One can type the keyword or a sub-string of a keyword in the text field and click on "Find" button on the *Find* window, and the tool will highlight the found key word. If one clicks on the "Find" button again, the next keyword if any will be highlighted and so on.

Figure 4-5: the Find window

Clicking on the "Close" button on the GUI of this tool will close the GUI and the tool. If selection changes have happened since the last "save", a `Close` window will pops up, as shown in figure 4-6, to remind the user to save the changes.

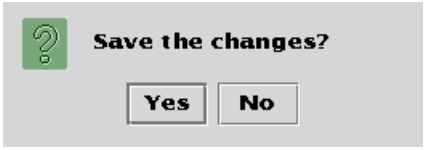

Figure 4-6: the Close Window

4.5 The Implementation of This Application

4.5.1 Overview

We chose Java as the programming language in the development of this Notation Selection Tool, as it is becoming a popular programming language in developing GUI applications and has good portability.

There are 15 classes and interfaces developed for this tool. We have created a package, named MathTools. Packages are just groups of Java classes and interfaces that are related by purpose or by application. They provide a means for organizing Java classes and interfaces. All the 15 classes and interfaces are grouped into the package MathTools.

The architecture of the package can be overviewed in the class-hierarchy and interface-hierarchy generated by *javadoc* utility.

Class Hierarchy

- class java.lang.Object
  - class java.util.AbstractCollection (implements java.util.Collection)
    - class java.util.AbstractList (implements java.util.List)
      - class java.util.AbstractSequentialList
        - class java.util.LinkedList (implements java.lang.Cloneable, java.util.List, java.io.Serializable)
          - class **SortedList**
  - class **Catalog** (implements Comparable)
  - class **Choice**
  - class java.awt.Component (implements java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable)
    - class java.awt.Container
      - class javax.swing.JComponent (implements java.io.Serializable)
        - class javax.swing.JPanel (implements javax.accessibility.Accessible)
          - class **CatalogPanel**
            - class **ComboBoxCatalogPanel**
            - class **RadioButtonCatalogPanel**
          - class **ItemPanel**
            - class **ComboBoxItemPanel** (implements java.awt.event.ActionListener)
            - class **RadioButtonItemPanel** (implements java.awt.event.ActionListener)
      - class java.awt.Window
        - class java.awt.Frame (implements java.awt.MenuContainer)
          - class javax.swing.JFrame (implements javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants)
            - class **FindFrame** (implements java.awt.event.ActionListener)
            - class **MathNotationsFrame** (implements java.awt.event.ActionListener)
  - class **DataReader**
  - class **Item** (implements Comparable)
  - class **NotationSelection**
  - class **Selecter**

Interface Hierarchy

- interface **Comparable**

4.5.2 The Choice class

The `Choice` class is designed for storing the data of a Choice element in the input XML document. The relationship between A `Choice` class object, a CHOICE element, and a notation is one-to-one. The `Choice` class has the following attributes: *image* of type `String` which is the file name of this notation's image, *keyvalue* of type `String` which is the ID of this notation, *presentation* of type `Element`(XML Element) which contains the templates for transformation to this notation. The `Choice` class has functions to get or set its attributes.

4.5.3 The Item class

The class `Item` is designed for mathematical items, such as DIVISION and MULTIPLICATION, and for storing the data of an ITEM element of the input XML document.

It has *title*, *keyword, choicelist, selection, selectedPresentation* attributes. The *title* attribute is the title of this item. The *keyword* attribute is the ID of this item. The *choicelist* is a linked list of Choice objects.The *selection* is the *keyvalue* of the selected notation for this item, and *selectedPresentation* is the PRESENTATION `Element` for this selected notation in the input XML document.

4.5.4 The Catalog class

The `Catalog` class is designed for storing the data of the CATALOG elements of the input XML file. A `Catalog` object corresponds to a mathematical group, such as ARITHMETIC and TRIGONOMETRY. It has *name* and *itemList* attributes. The *name* attribute is the name of this Catalog, and the *itemList* is a linked list of `Item`s in this `Catalog`.

4.5.5 The ItemPanel class

The `ItemPanel` class is a subclass of `JPanel`, which is a Swing component class. It defines the common behaviors of its subclasses, `RadioButtonItemPanel` class and `ComboBoxItemPanel` class. An `ItemPanel` object associates with an `Item` object. It displays the keyword and notation choices of the `Item` object. The `ItemPanel` class has two attributes: *item* of type `Item` corresponding to the `ItemPanel` object, and *textField* of type `JTextField` which shows the keyword of the `Item` object. It has an abstract method *getTextField()*, which is the common operation of its subclasses for accessing the *textField.*

4.5.6 The ComboBoxItemPanel class

The `ComboBoxItemPanel` class is a subclass of `ItemPanel`, and it implements the interface `ActionListener` in Swing. A `ComboBoxItemPanel` object contains two components: a `JTextField` showing the keyword and a dropdown `JComboBox` containing the list of notations of the corresponding `Item`. So the notation choices of an item are in a dropdown box. The `ComboBoxItemPanel` class has four methods, *addItemText()*, *addChoiceBox()*, *getTextField()*, and *actionPerformed()*. The

addItemText() creates a `JtextField` component containing the keyword and adds it to this panel. The *addChoiceBox()* creates the JComboBox component and adds it to this panel. The *getTextField()*returns the text field, and *actionPerformed()* implements the interface `ActionListener`.

4.5.7 The RadioButtonItemPanel class

The `RadioButtonItemPanel` class is a subclass of `ItemPanel`, and it implements the interface `ActionListener` in Swing. A `RadioButtonItemPanel` object contains a text field and a group of radio buttons which displays a list of notations of the corresponding `Item`. So the notation choices of an item are in a group of `JRadioButtons`. The `RadioButtonItemPanel` class has four methods, *addTextField()*, *addChoiceButtons()*, *getTextField()*, and *actionPerformed()*. The *addTextField()* creates a `JtextField` component containing the keyword and adds it to this panel. The *addChoiceButtons()* creates a group of `JRadioButton` components and adds them to this panel. The *getTextField()*returns the text field, and *actionPerformed()* implements the interface `ActionListener`.

4.5.8 The CatalogPanel class

The `CatalogPanel` class is a subclass of `JPanel`, which is a Swing component class. It defines the common behaviors of its subclasses, `RadioButtonCatalogPanel` class and `ComboBoxCatalogPanel` class. A `CatalogPanel` object associates with an `Catalog` object and contains a list of `ItemPanel` objects. `CatalogPanel` has an abstract method *addItemPanels()*, which is the common operation of its subclasses for creating and adding the *`ItemPanel`*s.

4.5.9 The ComboBoxCatalogPanel class

The `ComboBoxCatalogPanel` class is a subclass of `CatalogPanel`. A `ComboBoxCatalogPanel` object contains a list of `ComboBoxItemPanel` objects. `ComboBoxCatalogPanel` implements the methods, *addItemPanels()*, in its super class `CatalogPanel`.

4.5.10 The RadioButtonCatalogPanel class

The `RadioButtonCatalogPanel` class is a subclass of `CatalogPanel`, A `RadioButtonCatalogPanel` object contains a list of `RadioButtonItemPanel` objects. `RadioButtonCatalogPanel` implements the methods, *addItemPanels()*, of its super class `CatalogPanel`.

4.5.11 The FindFrame class

The `FindFrame` class is a subclass of `JFrame` in the `Swing` package and implements the interface `ActionListener`. It was designed to let the user to find a keyword in order, which has the input string as its sub-string. A `FindFrame` object is generated when the user clicking on the "Find" button on the GUI of the Notation Selection Tool. A `FindFrame` consists of a `JTextField` and three `JButton` objects labeled "Find", "Clear", and "Close" respectively. The `JTextField` lets the user type in the keyword or a sub-string of a keyword to be found. When the user clicks on the "Find" button, the keyword found is highlighted, or a `JOptionPane` is popped up telling the user that the

keyword is not found. If the user clicks on " Find" button again, the next keyword will be highlighted if any, and so on. When the user clicks on the "Clear" button, the `JTextField` will be cleared. Clicking on the "Close" button will close the `FindFrame` window.

4.5.12 The MathNotationsFrame class

Class `MathNotationsFrame` is a subclass of `Jframe`. It implements the `ActionListener` interface.

A `MathNotationsFrame` object is a GUI for the Notation Selection Tool. It displays all the selected mathematical items with their notations. The items are grouped into catalogs. It has two styles: RadioButton style and Dropdown style. This interface also provides some functional buttons for the user to load the saved choices, save their choices, find some math items, display the math objects, or close the interface. It can also let the user to specify the input and output files.

A `MathNotationsFrame` contains three `JPanel` objects. The top `JPanel` contains a `JTabbedPane`, which shows a group of math items and their notations at a time. The middle `JPanel` contains the a group of two `JRadioButton` objects which let the user select the style of the GUI, i. e. RadioButton style or Dropdown style, and four `JTextField` objects which let the user to specify the input and output files. The bottom `JPanel` contains the functional buttons labeled "Load", "Save", "Find", "Display", and "Close" respectively. Clicking on the "Load" or "Save" button will load the saved notation choices if any, or save the currently selected choices respectively. Either the key values or the XSLT style sheet can be saved, or both of them can be saved. When the user clicks on the "Find" button, a `FindFrame` is popped up, and the user can type in a sub-string of a keyword and the keyword will be highlighted on the GUI if any. When the user clicks on the "Display" button, the input content MathML markups might be

transformed to present MathML markups and then the mathematical objects can be rendered with the selected notations. Clicking on the "Close" button will close the GUI.

The `MathNotationsFrame` class has the following important methods:

actionPerformed(java.awt.event.ActionEvent evt) implements the interface `ActionListener`. When the user clicks on the different buttons, it performs different functionality.

createTabbedPane() creates the bottom panel of this frame.

CreateCBTabbedPane() creates a `JTabbedPane` in which the notations displayed as drop down box.

createMiddlePanel () creates the middle panel of this frame.

createRBTabbedPane() creates a `JTabbedPane` in which the notations displayed as radio buttons.

createTopPanel() creates the top panel which contains all math Items and their notations.

disposeFindFrame() sets the findFrame attribute to be null when the `FindFrame` window is closed.

getCatalogList() gets the list of Catalogs for the `FindFrame` class to use.

getTabbedPane() gets the `jtp` attribute which is a `JTabbedPane` contained in the top panel of this frame for the `FindFrame` class to use.

GetTop() gets the top panel of this frame for the `FindFrame` class to use.

loadChoices() loads the saved choices from file "Choice.dat".

saveChoices() saves the key values of selected notations.

saveStyleSheet() appends the templates for the selected notations to the input style sheet and saves as a new stylesheet for the transformation.

4.5.13 The NotationSelection class

`NotationSelection` class is the engine class, which contains the main method for the Notation Selection Tool.

4.5.14 The DataReader class

The `DataReader` class is designed for extracting data from a parsed XML tree.

A `DataReader` object extracts the data from a parsed XML tree and constructs a `SortedList` of `Catalog` objects.

`DataReader` has the following methods:

getCatalogs() returns the constructed `SortedList` of `Catalog` objects;

*readCatalog(Node node)*reads a CATALOG node in the XML document and constructs a `Catalog` object;

readItem(Node node) reads an ITEM node in the XML document and constructs an `Item` object;

readTitle(Node node) reads the title of an `Item`;

readKeyword(Node node) reads the keyword of an `Item`;

readChoiceList(Node node) reads the `Choice` list of an Item;

readChoice(Node node) reads a `Choice` of an Item;

readImage(Node node) reads the image file name of a `Choice`;

readKeyvalue(Node node) reads the *Keyvalue* of a `Choice`.

4.5.15 The SortedList class

A SortedList is a list of Comparable objects in ascending order. The `SortedList` is a subclass of LinkedList in the Java `util` package. `SortedList` has three methods, *insert(Comparable o)* which inserts a Comparable object into the list, *insertAfter(int here, Comparable o)* which is a helper method of *insert* and begin from the index *here* in the list to recursively find the right position and insert the comparable object in the list, and *displayAll()* which displays all the objects in the list.

4.5.16 The Comparable Interface

The `Comparable` interface has a method compareTo*(Object compareMe)* which compares the object implementing interface Comparable to *compareMe*, and returns an

integer less than 0 if compareMe is bigger than the object, 0 if compareMe is indistinguishable from the object , and an integer greater than 0 if compareMe is smaller than the object.

In MathTools package, the `Item` class and `Catalog` class implement interface `Comparable` in order to sort all `Item` objects and `Catalog` objects alphabetically.

5. Conclusion

We have developed a notation selection tool for the content-to-presentation transformation of MathML. This tool provides the user with a graphical user interface, which allows the user to select his/her notation conventions while transforming Content MathML to Presentation MathML. We use an XML document to organize the mathematical notations so that this notation selection tool is extensible. In this project, we have selected a realistic set of mathematical notations with which this tool can provides the user. With the help of other tools, such as the XSLT processor XT and the mathematical Viewer developed by Luca Padovani, we have tested this notation selection tool by selecting different notations. The testing shows that this tool works properly for all the selected notations.

References

1. W3C Recommendation: *Mathematical Markup Language (MathML) 1.01 specification* http://www.w3.org/TR/REC-MathML/

2. W3C Recommendation: *Mathematical Markup Language (MathML) Version 2.0* http://www.w3.org/TR/MathML2/

3. W3C Recommendation: *Extensible Markup Language(XML) 1.0,* http://www.w3.org/TR/2000/REC-xml-20001006

4. Igor Rodionov : mml2mml.xsl --XSL Transformation of MathML content to MathML presentation.

5. W3C Recommendation: *XSL Transformations (XSLT) Version 1.0* http://www.w3.org/TR/xslt.html

6. Xuehong Li. *XML and the COMMUNICATION OF MATHEMATICAL OBJECTS*

7. Neil Brandley. *The XML companion. Second Edition*

8. Kathy Walrath, Mary Campione. *The JFC Swing tutorial : a guide to constructing GUIs*

9. Java ™ 2 Platform Standard Edition: http://java.sun.com/products/jdk/1.2/docs/api/index.htm

9. Xerces API*: http://xml.apache.org/apiDocs/index.html*

Appendix A – The Input XML Document

```
<?xml version="1.0"?>

<!-- ************************************************************
 * mnotation.mss:  the input XML file for organizing different
 *                 mathematical notations
 * Project: Notation Selection Tools for MathML Style Sheets.
 * @author: Dicheng Liu
 * Dept of Computer Science, the university of western Ontario
 * Fall, 2000
 ************************************************************ -->

<mnotations>

<!-- ************** ARITHMETIC ***********************  -->
  <catalog>
    <name>Arithmetic</name>
    <itemlist>

      <item>
        <title> continued fraction </title>
        <content>
          <!-- ???? →
        </content>
        <keyword> CONTINUED FRACTION </keyword>
        <choicelist>

          <!-- *************CONTINUED FRACTION CHOICE 1************** -->
          <choice>
            <image src = "cf1.gif"/>
          <keyvalue> 1 </keyvalue>
            <presentation>
              <xsl:template match = "mml:apply[*[1][self::mmlx:contfrac]]"
                            name="contfrac" priority="100">
                <xsl:param name="pos" select="2"/>
                <mrow>
                  <xsl:apply-templates select="*[$pos]"/>
                  <mo>+</mo>

                  <xsl:choose>
                    <xsl:when test="*[last()=$pos]">
                      <mo>...</mo>
                    </xsl:when>
                    <xsl:otherwise>
                      <mfrac>
                        <mn>1</mn>
                        <xsl:call-template name="contfrac">
                          <xsl:with-param name="pos" select="$pos+1"/>
                        </xsl:call-template>
                      </mfrac>
                    </xsl:otherwise>
                  </xsl:choose>
                </mrow>
              </xsl:template>
```

```
      </presentation>
    </choice>

    <!-- *************CONTINUED FRACTION CHOICE 2 ************** -->
    <choice>
      <image src = "cf2.gif"/>
    <keyvalue> 2 </keyvalue>
      <presentation>
        <xsl:template match = "mml:apply[*[1][self::mmlx:contfrac]]"
                      priority="100">
          <mrow>
            <mfenced open="[" close="]">
              <xsl:apply-templates select = "*" mode = "semantics"/>
              <mi>...</mi>
            </mfenced>
          </mrow>
        </xsl:template>
      </presentation>
    </choice>

    <!-- *************CONTINUED FRACTION CHOICE 3 ************** -->
    <choice>
      <image src = "cf3.gif"/>
    <keyvalue> 3 </keyvalue>
      <presentation>

        <xsl:template match = "mml:apply[*[1][self::mmlx:contfrac]]"
                      priority="100">
          <mrow>
            <xsl:apply-templates select = "*[2]" mode = "semantics"/>
            <mo>+</mo>
            <xsl:for-each select="*[position()>2]">
              <mfrac>
                <mrow>
                  <mn>1</mn>
                  <mo rspace='-5 px' lspace='5 px'>|</mo>
                </mrow>
                <mrow>
                  <mo lspace='0' rspace='5 px'>|</mo>
                  <mn><xsl:apply-templates/></mn>
                </mrow>
              </mfrac>
              <mo lspace='5 px' rspace='5 px'>+</mo>
            </xsl:for-each>
            <mi>...</mi>
          </mrow>
        </xsl:template>
      </presentation>
    </choice>
  </choicelist>
</item>

<!--   ******* DIVISION ********************** -->
<item>
  <title> x divided by y  </title>
  <content>
    <!-- ???? →
```

```
</content>
<keyword> DIVISION </keyword>
<choicelist>
  <!--  *********** DIVISION  CHOICE 1  ********** -->
  <choice>
    <image src = "div1.gif"/>
    <keyvalue> 1 </keyvalue>
    <presentation>
      <xsl:template match = "mml:apply[ mml:divide[1]]"
                    mode="remdiv" priority = "100">
        <xsl:param name="PARAM" select="$NO_PARAM"/>
        <xsl:param name="PAREN" select="$PAR_NO"/>
        <xsl:param name="PAR_NO_IGNORE" select="$YES"/>
        <mrow>
        <xsl:apply-templates select = "*[2]" mode = "semantics">
          <xsl:with-param name="IN_PREC" select="$DIV_PREC"/>
          <xsl:with-param name="PARAM" select="$PARAM"/>
          <xsl:with-param name="PAREN" select="$PAREN"/>
          <xsl:with-param name="PAR_NO_IGNORE" select="$PAR_NO_IGNORE"/>
        </xsl:apply-templates>
        <mo><mchar name="divide"/></mo>
        <xsl:apply-templates select = "*[3]" mode = "semantics">
          <xsl:with-param name="IN_PREC" select="$DIV_PREC"/>
          <xsl:with-param name="PARAM" select="$PAR_SAME"/>
          <xsl:with-param name="PAREN" select="$PAREN"/>
          <xsl:with-param name="PAR_NO_IGNORE" select="$NO"/>
        </xsl:apply-templates>
        </mrow>
      </xsl:template>
    </presentation>
  </choice>
  <!--  *********** DIVISION  CHOICE 2  ********** -->
  <choice>
    <image src = "div2.gif"/>
    <keyvalue> 2 </keyvalue>
    <presentation>
      <xsl:template match = "mml:apply[mml:divide[1]]"
                    mode="remdiv" priority = "100">
        <xsl:param name="PARAM" select="$NO_PARAM"/>
        <xsl:param name="PAREN" select="$PAR_NO"/>
        <xsl:param name="PAR_NO_IGNORE" select="$YES"/>
        <mrow>
        <xsl:apply-templates select = "*[2]" mode = "semantics">
          <xsl:with-param name="IN_PREC" select="$DIV_PREC"/>
          <xsl:with-param name="PARAM" select="$PARAM"/>
          <xsl:with-param name="PAREN" select="$PAREN"/>
          <xsl:with-param name="PAR_NO_IGNORE"
                          select="$PAR_NO_IGNORE"/>
        </xsl:apply-templates>
        <mo>/</mo>
        <xsl:apply-templates select = "*[3]" mode = "semantics">
          <xsl:with-param name="IN_PREC" select="$DIV_PREC"/>
          <xsl:with-param name="PARAM" select="$PAR_SAME"/>
          <xsl:with-param name="PAREN" select="$PAREN"/>
          <xsl:with-param name="PAR_NO_IGNORE" select="$NO"/>
        </xsl:apply-templates>
        </mrow>
```

```
    </xsl:template>
  </presentation>
</choice>

<!--  *********** DIVISION  CHOICE 3  ********** -->
<choice>
  <image src = "div3.gif"/>
  <keyvalue> 3 </keyvalue>
  <presentation>
    <xsl:template match = "mml:apply[ mml:divide[1]]"
                  mode="remdiv"  priority = "100">
      <xsl:param name="PARAM" select="$NO_PARAM"/>
      <xsl:param name="PAREN" select="$PAR_NO"/>
      <xsl:param name="PAR_NO_IGNORE" select="$YES"/>
      <mfrac>
      <xsl:apply-templates select = "*[2]" mode = "semantics">
        <xsl:with-param name="IN_PREC" select="$DIV_PREC"/>
        <xsl:with-param name="PARAM" select="$PARAM"/>
        <xsl:with-param name="PAREN" select="$PAREN"/>
        <xsl:with-param name="PAR_NO_IGNORE"
                        select="$PAR_NO_IGNORE"/>
      </xsl:apply-templates>
      <xsl:apply-templates select = "*[3]" mode = "semantics">
        <xsl:with-param name="IN_PREC" select="$DIV_PREC"/>
        <xsl:with-param name="PARAM" select="$PAR_SAME"/>
        <xsl:with-param name="PAREN" select="$PAREN"/>
        <xsl:with-param name="PAR_NO_IGNORE" select="$NO"/>
      </xsl:apply-templates>
      </mfrac>
    </xsl:template>
  </presentation>
</choice>

<!--  *********** DIVISION  CHOICE 4  ********** -->
<choice>
  <image src = "div4.gif"/>
  <keyvalue> 4 </keyvalue>
  <presentation>
    <xsl:template match = "mml:apply[mml:divide[1]]"
                  mode="remdiv" priority = "100">
      <xsl:param name="PARAM" select="$NO_PARAM"/>
      <xsl:param name="PAREN" select="$PAR_NO"/>
      <xsl:param name="PAR_NO_IGNORE" select="$YES"/>
      <xsl:apply-templates select = "*[3]" mode = "semantics">
        <xsl:with-param name="IN_PREC" select="$DIV_PREC"/>
        <xsl:with-param name="PARAM" select="$PARAM"/>
        <xsl:with-param name="PAREN" select="$PAREN"/>
        <xsl:with-param name="PAR_NO_IGNORE"
                        select="$PAR_NO_IGNORE"/>
      </xsl:apply-templates>

      <menclose notation='longdiv'>
      <xsl:apply-templates select = "*[2]" mode = "semantics">
        <xsl:with-param name="IN_PREC" select="$DIV_PREC"/>
        <xsl:with-param name="PARAM" select="$PAR_SAME"/>
        <xsl:with-param name="PAREN" select="$PAREN"/>
        <xsl:with-param name="PAR_NO_IGNORE" select="$NO"/>
```

```
          </xsl:apply-templates>
          </menclose>
        </xsl:template>
      </presentation>
    </choice>
  </choicelist>
</item>

<!-- ******* MULTIPLICATION ******************  -->
<item>
  <title>x multiply y </title>
  <content>
    <!-- ???? →
  </content>
  <keyword> MULTIPLICATION </keyword>
  <choicelist>

    <!-- ******* MULTIPLICATION  CHOICE 1 *********  -->
    <choice>
      <image src = "mult1.gif"/>
      <keyvalue> 1 </keyvalue>
      <presentation>

        <xsl:template match = "mml:apply[mml:times[1]]" mode="times"
                      priority = "100">
          <xsl:param name="PARAM" select="$NO_PARAM"/>
          <xsl:param name="PAREN" select="$PAR_NO"/>
          <xsl:param name="PAR_NO_IGNORE" select="$YES"/>
          <xsl:apply-templates select="*[2]" mode = "semantics">
            <xsl:with-param name="IN_PREC" select="$MUL_PREC"/>
            <xsl:with-param name="PARAM" select="$PARAM"/>
            <xsl:with-param name="PAREN" select="$PAREN"/>
            <xsl:with-param name="PAR_NO_IGNORE"
                            select="$PAR_NO_IGNORE"/>
          </xsl:apply-templates>
          <xsl:if test="*[3]">
            <xsl:for-each select = "*[position()>2]">
              <mo><mchar name="times"/> </mo>
              <xsl:apply-templates select="." mode = "semantics">
                <xsl:with-param name="IN_PREC" select="$MUL_PREC"/>
                <xsl:with-param name="PAREN" select="$PAREN"/>
                <xsl:with-param name="PAR_NO_IGNORE" select="$NO"/>
              </xsl:apply-templates>
            </xsl:for-each>
          </xsl:if>
        </xsl:template>
      </presentation>
    </choice>

    <!-- ******* MULTIPLICATION  CHOICE 2 *********  -->
    <choice>
      <image src = "mult2.gif"/>
      <keyvalue> 2 </keyvalue>
      <presentation>
        <xsl:template match = "mml:apply[mml:times[1]]" mode="times"
                      priority = "100">
          <xsl:param name="PARAM" select="$NO_PARAM"/>
```

```
            <xsl:param name="PAREN" select="$PAR_NO"/>
            <xsl:param name="PAR_NO_IGNORE" select="$YES"/>
            <xsl:apply-templates select="*[2]" mode = "semantics">
              <xsl:with-param name="IN_PREC" select="$MUL_PREC"/>
              <xsl:with-param name="PARAM" select="$PARAM"/>
              <xsl:with-param name="PAREN" select="$PAREN"/>
              <xsl:with-param name="PAR_NO_IGNORE"
                              select="$PAR_NO_IGNORE"/>
            </xsl:apply-templates>
            <xsl:if test="*[3]">
              <xsl:for-each select = "*[position()>2]">
                <mo><mchar name="middot"/></mo>
                <xsl:apply-templates select="." mode = "semantics">
                  <xsl:with-param name="IN_PREC" select="$MUL_PREC"/>
                  <xsl:with-param name="PAREN" select="$PAREN"/>
                  <xsl:with-param name="PAR_NO_IGNORE" select="$NO"/>
                </xsl:apply-templates>
              </xsl:for-each>
            </xsl:if>
          </xsl:template>
        </presentation>
      </choice>

      <!-- ******* MULTIPLICATION  CHOICE 3 *********  -->
      <choice>
        <image src = "mult3.gif"/>
        <keyvalue> 3 </keyvalue>
        <presentation>
          <xsl:template match = "mml:apply[mml:times[1]]" mode="times"
                    priority = "100">
            <xsl:param name="PARAM" select="$NO_PARAM"/>
            <xsl:param name="PAREN" select="$PAR_NO"/>
            <xsl:param name="PAR_NO_IGNORE" select="$YES"/>
            <xsl:apply-templates select="*[2]" mode = "semantics">
              <xsl:with-param name="IN_PREC" select="$MUL_PREC"/>
              <xsl:with-param name="PARAM" select="$PARAM"/>
              <xsl:with-param name="PAREN" select="$PAREN"/>
              <xsl:with-param name="PAR_NO_IGNORE"
                              select="$PAR_NO_IGNORE"/>
            </xsl:apply-templates>
            <xsl:if test="*[3]">
              <xsl:for-each select = "*[position()>2]">
                <mo> <mchar name="InvisibleTimes"/> </mo>
                <xsl:apply-templates select="." mode = "semantics">
                  <xsl:with-param name="IN_PREC" select="$MUL_PREC"/>
                  <xsl:with-param name="PAREN" select="$PAREN"/>
                  <xsl:with-param name="PAR_NO_IGNORE" select="$NO"/>
                </xsl:apply-templates>
              </xsl:for-each>
            </xsl:if>
          </xsl:template>
        </presentation>
      </choice>
    </choicelist>
  </item>
 </itemlist>
</catalog>
```

```
<!-- *************** TRIG ******************************** -->

  <catalog>
    <name>Trig</name>
    <itemlist>
      <!-- *******  TANGENT ********************** -->
      <item>
        <title> tangent of x </title>
        <content>
          <!-- ???? →
        </content>
        <keyword> TANGENT </keyword>
        <choicelist>

          <!-- *******  TANGENT CHOICE 1 ************** -->
          <choice>
            <image src = "tan1.gif"/>
          <keyvalue> 1 </keyvalue>
            <presentation>
               <xsl:template match = "mml:tan" mode = "trigonometry"
                             priority="100">
                  <mo>tan</mo>
             </xsl:template>
            </presentation>
          </choice>

          <!-- *******  TANGENT CHOICE 2 ************** -->
          <choice>
            <image src = "tan2.gif"/>
          <keyvalue> 2 </keyvalue>
            <presentation>
               <xsl:template match = "mml:tan" mode = "trigonometry"
                             priority="100">
                  <mo>tg</mo>
             </xsl:template>
            </presentation>
          </choice>
        </choicelist>
      </item>

      <!-- ******* COTANGENT **************  -->
      <item>
        <title> cotangent of x </title>
        <content>
          <!-- ???? →
        </content>
        <keyword> COTANGENT </keyword>
        <choicelist>

          <!-- ******* COTANGENT CHOICE 1 **************  -->
          <choice>
            <image src = "ctan1.gif"/>
          <keyvalue> 1 </keyvalue>
            <presentation>
               <xsl:template match = "mml:cot" mode = "trigonometry"
```

```
                    priority="100">
            <mo>cotan</mo>
       </xsl:template>

      </presentation>
    </choice>

    <!-- ******* COTANGENT CHOICE 2 ***************  -->
    <choice>
      <image src = "ctan2.gif"/>
    <keyvalue> 2 </keyvalue>
      <presentation>
         <xsl:template match = "mml:cot" mode = "trigonometry"
                    priority="100">
            <mo>ctg</mo>
       </xsl:template>

      </presentation>
    </choice>

    <!-- ******* COTANGENT CHOICE 3 ***************  -->
    <choice>
      <image src = "ctan3.gif"/>
    <keyvalue> 3 </keyvalue>
      <presentation>
         <xsl:template match = "mml:cot" mode = "trigonometry"
                    priority="100">
            <mo>cot</mo>
       </xsl:template>

      </presentation>
    </choice>
  </choicelist>
</item>

<!-- ******** ARCSINE ************************ -->
<item>
  <title> arc sin of x </title>
  <content>
    <!-- ???? →
  </content>
  <keyword> ARCSIN </keyword>
  <choicelist>

    <!-- ******** ARCSINE CHOICE 1 ************************ -->
    <choice>
      <image src = "arcsin1.gif"/>
    <keyvalue> 1 </keyvalue>
      <presentation>
         <xsl:template match = "mml:arcsin" mode = "trigonometry"
              priority="100">
            <msup>
              <mo>sin</mo>
              <mn>-1</mn>
            </msup>
       </xsl:template>
```

```
          </presentation>
        </choice>

        <!-- ******** ARCSINE CHOICE 2 ********************** -->
        <choice>
          <image src = "arcsin2.gif"/>
        <keyvalue> 2 </keyvalue>
          <presentation>
             <xsl:template match = "mml:arcsin" mode = "trigonometry"
                          priority="100">
                <mo>arcsin</mo>
           </xsl:template>
          </presentation>
        </choice>

        <!-- ******** ARCSINE CHOICE 3 ********************** -->
        <choice>
          <image src = "arcsin3.gif"/>
        <keyvalue> 3 </keyvalue>
          <presentation>
             <xsl:template match = "mml:arcsin" mode = "trigonometry"
                          priority="100">
                <mo>asin</mo>
           </xsl:template>

          </presentation>
        </choice>
      </choicelist>
    </item>
  </itemlist>
</catalog>

<!-- **************POWER OR ROOT *************** -->

  <catalog>
    <name>Power or Root</name>
    <itemlist>

      <!-- ******* EXPONENT ********************************** -->
      <item>
        <title> exponent of x </title>
        <content>
          <!-- ???? →
        </content>
        <keyword> EXPONENT </keyword>
        <choicelist>

          <!-- ******* EXPONENT CHOICE 1 ********************** -->
          <choice>
            <image src = "exp1.gif"/>
          <keyvalue> 1 </keyvalue>
            <presentation>
               <xsl:template match = "mml:apply[*[1][self::mml:exp]]"
                          priority="100">
                 <mrow>
                   <mo>exp</mo>
                   <mfenced>
```

```
            <xsl:apply-templates select = "*[2]" mode = "semantics"/>
            </mfenced>
          </mrow>
        </xsl:template>
      </presentation>
    </choice>

    <!-- ******* EXPONENT CHOICE 2 ********************** -->
    <choice>
      <image src = "exp2.gif"/>
    <keyvalue> 2 </keyvalue>
      <presentation>
        <xsl:template match = "mml:apply[*[1][self::mml:exp]]"
                      priority="100">
          <msup>
            <mo>e</mo>
            <xsl:apply-templates select = "*[2]" mode = "semantics"/>
          </msup>
        </xsl:template>
      </presentation>
    </choice>
  </choicelist>
 </item>

<!--  ************* ROOT ******************************** -->
 <item>
   <title> the n-th root of x </title>
   <content>
     <!-- ???? →
   </content>
   <keyword> ROOT </keyword>
   <choicelist>

     <!-- ******** ROOT CHOICE 1 ********************** -->
     <choice>
       <image src = "root1.gif"/>
     <keyvalue> 1 </keyvalue>
       <presentation>
         <xsl:template match = "mml:apply[mml:root[1]]" priority="100">
              <xsl:if test="*[2]=mml:degree">
               <mroot>
               <xsl:apply-templates select="*[3]" mode = "semantics"/>
               <xsl:apply-templates select="*[2]" mode = "semantics"/>
               </mroot>
             </xsl:if>
             <xsl:if test="not(*[2]=mml:degree)">
               <msqrt>
               <xsl:apply-templates select="*[2]" mode = "semantics"/>
               <mn>2</mn>
               </msqrt>
             </xsl:if>
          </xsl:template>
       </presentation>
     </choice>

     <!-- ******** ROOT CHOICE 2 ********************** -->
     <choice>
```

```
          <image src = "root2.gif"/>
        <keyvalue> 2 </keyvalue>
          <presentation>

             <xsl:template match = "mml:apply[mml:root[1]]" priority="100">
               <msup>
                 <xsl:if test="*[2]=mml:degree">
                   <xsl:apply-templates select="*[3]" mode = "semantics"/>
                   <mrow>
                     <mn>1</mn>
                     <mo>/</mo>
                     <xsl:apply-templates select="*[2]" mode = "semantics"/>
                   </mrow>
                 </xsl:if>
                 <xsl:if test="not(*[2]=mml:degree)">
                   <xsl:apply-templates select="*[2]" mode = "semantics"/>
                   <mrow>
                     <mn>1</mn>
                     <mo>/</mo>
                     <mn>2</mn>
                   </mrow>
                 </xsl:if>
               </msup>
             </xsl:template>
          </presentation>
        </choice>
      </choicelist>
    </item>

  </itemlist>
</catalog>


<!-- ************** LOGARITHM ***************** -->
<catalog>
  <name>Logarithm</name>
  <itemlist>

    <!-- ********* BASE 2 *****************-->
    <item>
      <title> logarithms </title>
      <content>
        <!-- ???? →
      </content>
      <keyword> LOG BASE 2 </keyword>
      <choicelist>

        <!-- ********* BASE 2 CHOICE 1 *****************-->
        <choice>
          <image src = "base_2_1.gif"/>
        <keyvalue> 1 </keyvalue>
          <presentation>
            <xsl:template match = "mml:apply[*[1][self::mml:log] and
                         *[2][self::mml:logbase]]" priority='101'>

              <xsl:variable name="m">
                <xsl:value-of select="*[2]/*[1]"/>
```

```
      </xsl:variable>
      <mrow>
        <xsl:choose>
          <xsl:when test="parent::mml:apply[mml:power[1]]">
            <xsl:if test="$m='2'">
              <msup>
                <mo>lg</mo>
                <xsl:apply-templates select = "../*[3]"
                                     mode = "semantics"/>
              </msup>
            </xsl:if>
            <xsl:if test="not ($m='2')">
              <msup>
                <mo>log</mo>
                <xsl:apply-templates select = "../*[3]"
                                     mode = "semantics"/>
                <xsl:apply-templates select="mml:logbase"
                                     mode="semantics"/>
              </msup>
            </xsl:if>
          </xsl:when>
          <xsl:otherwise>
            <xsl:if test="$m='2'">
              <mo>lg</mo>
            </xsl:if>
            <xsl:if test="not ($m='2')">
              <msub>
                <mo>log</mo>
                <xsl:apply-templates select="mml:logbase"
                                     mode = "semantics"/>
              </msub>
            </xsl:if>
          </xsl:otherwise>
        </xsl:choose>
        <mo> <mchar name='ApplyFunction'/> </mo>
          <xsl:apply-templates select = "*[3]" mode = "semantics">
            <xsl:with-param name="IN_PREC" select="$FUNCTION_PREC"/>
          </xsl:apply-templates>
      </mrow>
    </xsl:template>
  </presentation>
</choice>

<!-- ********* BASE 2 CHOICE 2 *****************-->
<choice>
  <image src = "base_2_2.gif"/>
<keyvalue> 2 </keyvalue>
  <presentation>
    <xsl:template match = "mml:apply[*[1][self::mml:log] and
                  *[2][self::mml:logbase]]"  priority='101'>
      <mrow>
        <xsl:choose>
          <xsl:when test="parent::mml:apply[mml:power[1]]">
              <msubsup>
                <mo>log</mo>
                <xsl:apply-templates select = "../*[3]"
                                     mode = "semantics"/>
```

```
                <xsl:apply-templates select="mml:logbase"
                                     mode="semantics"/>
              </msubsup>
            </xsl:when>
            <xsl:otherwise>
              <msub>
                <mo>log</mo>
                <xsl:apply-templates select="mml:logbase"
                                     mode="semantics"/>
              </msub>
            </xsl:otherwise>
          </xsl:choose>
          <mo> <mchar name='ApplyFunction'/> </mo>
            <xsl:apply-templates select = "*[3]" mode = "semantics">
              <xsl:with-param name="IN_PREC" select="$FUNCTION_PREC"/>
            </xsl:apply-templates>
        </mrow>
      </xsl:template>
    </presentation>
  </choice>

  <!-- ********* BASE 2 CHOICE 3 *****************-->
  <choice>
    <image src = "base_2_3.gif"/>
  <keyvalue> 3 </keyvalue>
    <presentation>
      <xsl:template match = "mml:apply[*[1][self::mml:log] and
                             *[2][self::mml:logbase]]" priority='101'>

        <xsl:variable name="m">
          <xsl:value-of select="*[2]/*[1]"/>
        </xsl:variable>
        <mrow>
          <xsl:choose>
            <xsl:when test="parent::mml:apply[mml:power[1]]">
              <xsl:if test="$m='2'">
                <msup>
                  <mo>log</mo>
                  <xsl:apply-templates select = "../*[3]"
                                       mode = "semantics"/>
                </msup>
              </xsl:if>
              <xsl:if test="not ($m='2')">
                <msup>
                  <mo>log</mo>
                  <xsl:apply-templates select = "../*[3]"
                                       mode = "semantics"/>
                  <xsl:apply-templates select="mml:logbase"
                                       mode="semantics"/>
                </msup>
              </xsl:if>
            </xsl:when>
            <xsl:otherwise>
              <xsl:if test="$m='2'">
                <mo>log</mo>
              </xsl:if>
              <xsl:if test="not ($m='2')">
```

```
                <msub>
                  <mo>log</mo>
                  <xsl:apply-templates select="mml:logbase"
                                       mode="semantics"/>
                </msub>
              </xsl:if>
            </xsl:otherwise>
          </xsl:choose>
          <mo> <mchar name='ApplyFunction'/> </mo>

            <xsl:apply-templates select = "*[3]" mode = "semantics">
              <xsl:with-param name="IN_PREC" select="$FUNCTION_PREC"/>
            </xsl:apply-templates>
        </mrow>
      </xsl:template>
    </presentation>
  </choice>
 </choicelist>
</item>

<!-- ************** BASE e ************************** -->

<item>
  <title> logarithms </title>
  <content>
    <!-- ???? →
  </content>
  <keyword> LOG BASE e </keyword>
  <choicelist>

    <!-- ************** BASE e CHOICE 1 ******************** -->
    <choice>
      <image src = "base_e1.gif"/>
    <keyvalue> 1 </keyvalue>
      <presentation>
        <xsl:template match = "mml:apply[*[1][self::mml:ln]]"
                      priority="100">
          <mrow>
            <xsl:choose>
              <xsl:when test="parent::mml:apply[mml:power[1]]">
                <msup>
                  <mo>ln</mo>
                  <xsl:apply-templates select = "../*[3]"
                                       mode = "semantics"/>
                </msup>
              </xsl:when>
              <xsl:otherwise>
                <mo>ln</mo>
              </xsl:otherwise>
            </xsl:choose>
            <mo> <mchar name='ApplyFunction'/> </mo>
            <xsl:apply-templates select = "*[2]" mode = "semantics">
              <xsl:with-param name="IN_PREC" select="$FUNCTION_PREC"/>
            </xsl:apply-templates>
          </mrow>
        </xsl:template>
      </presentation>
```

```
    </choice>

    <!-- ************** BASE e CHOICE 2  ******************** -->
    <choice>
      <image src = "base_e2.gif"/>
    <keyvalue> 2 </keyvalue>
      <presentation>
        <xsl:template match = "mml:apply[*[1][self::mml:ln]]"
                      priority="100">
          <mrow>
            <xsl:choose>
              <xsl:when test="parent::mml:apply[mml:power[1]]">
                <msup>
                  <mo>log</mo>
                  <xsl:apply-templates select = "../*[3]"
                                       mode = "semantics"/>
                </msup>
              </xsl:when>
              <xsl:otherwise>
                <mo>log</mo>
              </xsl:otherwise>
            </xsl:choose>
            <mo> <mchar name='ApplyFunction'/> </mo>
            <xsl:apply-templates select = "*[2]" mode = "semantics">
              <xsl:with-param name="IN_PREC" select="$FUNCTION_PREC"/>
            </xsl:apply-templates>
          </mrow>
        </xsl:template>
      </presentation>
    </choice>
  </choicelist>
</item>


<!-- **************BASE 10 ************************** -->

<item>
  <title> logarithms </title>
  <content>
    <!-- ???? →
  </content>
  <keyword> LOG BASE 10 </keyword>
  <choicelist>

    <!-- **************BASE 10 CHOICE 1 *************** -->
    <choice>
      <image src = "base_10_1.gif"/>
    <keyvalue> 1 </keyvalue>
      <presentation>
        <xsl:template match = "mml:apply[*[1][self::mml:log]  and
                      not (*[2][self::mml:logbase])]" priority='100'>
          <mrow>
            <xsl:choose>
              <xsl:when test="parent::mml:apply[mml:power[1]]">
                  <msubsup>
                    <mo>log</mo>
                    <xsl:apply-templates select = "../*[3]"
```

```
                                        mode = "semantics"/>
                        <mn>10</mn>
                      </msubsup>
                    </xsl:when>
                    <xsl:otherwise>
                      <msub>
                        <mo>log</mo>
                        <mn>10</mn>
                      </msub>
                    </xsl:otherwise>
                  </xsl:choose>
                  <mo> <mchar name='ApplyFunction'/> </mo>
                  <xsl:apply-templates select = "*[2]" mode = "semantics">
                    <xsl:with-param name="IN_PREC" select="$FUNCTION_PREC"/>
                  </xsl:apply-templates>
                </mrow>
              </xsl:template>
            </presentation>
          </choice>

          <!-- **************BASE 10 CHOICE 2 ********************** -->
          <choice>
            <image src = "base_10_2.gif"/>
          <keyvalue> 2 </keyvalue>
            <presentation>
              <xsl:template match = "mml:apply[*[1][self::mml:log]  and
                            not (*[2][self::mml:logbase])]" priority='100'>
                <mrow>
                  <xsl:choose>
                    <xsl:when test="parent::mml:apply[mml:power[1]]">
                      <msup>
                        <mo>log</mo>
                        <xsl:apply-templates select = "../*[3]"
                                        mode = "semantics"/>
                      </msup>
                    </xsl:when>
                    <xsl:otherwise>
                      <mo>log</mo>
                    </xsl:otherwise>
                  </xsl:choose>
                  <mo> <mchar name='ApplyFunction'/> </mo>
                  <xsl:apply-templates select = "*[2]" mode = "semantics">
                    <xsl:with-param name="IN_PREC" select="$FUNCTION_PREC"/>
                  </xsl:apply-templates>
                </mrow>
              </xsl:template>
            </presentation>
          </choice>
        </choicelist>
      </item>
    </itemlist>
  </catalog>

<!-- ************ COMBINATIONS ************** -->
  <catalog>
    <name>Combinations</name>
    <itemlist>
```

```
<item>
  <title> Combination </title>
  <content>
    <!-- ???? →
  </content>
  <keyword> COMBINATION </keyword>
  <choicelist>

    <!-- ************ COMBINATION CHOICE 1 ************** -->
    <choice>
      <image src = "comb1.gif"/>
    <keyvalue> 1 </keyvalue>
      <presentation>
        <xsl:template match = "mml:apply[*[1][self::mmlx:combination]]"
                      priority="100">
        <mrow>
          <mfenced>
            <mtable>
              <mtr>
                <xsl:apply-templates select = "*[2]" mode = "semantics"/>
              </mtr>
              <mtr>
                <xsl:apply-templates select = "*[3]" mode = "semantics"/>
              </mtr>
            </mtable>
          </mfenced>
        </mrow>
        </xsl:template>
      </presentation>
    </choice>

    <!-- ************ COMBINATION CHOICE 2 ************** -->
    <choice>
      <image src = "comb2.gif"/>
    <keyvalue> 2 </keyvalue>
      <presentation>
        <xsl:template match = "mml:apply[*[1][self::mmlx:combination]]"
                      priority="100">
        <mrow>
         <mmultiscripts>
          <mi fontweight="bold">C</mi>
          <xsl:apply-templates select = "*[3]" mode = "semantics"/>
          <xsl:apply-templates select = "*[2]" mode = "semantics"/>
         </mmultiscripts>
        </mrow>
        </xsl:template>
      </presentation>
    </choice>

    <!-- ************ COMBINATION CHOICE 3 ************** -->
    <choice>
      <image src = "comb3.gif"/>
    <keyvalue> 3 </keyvalue>
      <presentation>
        <xsl:template match = "mml:apply[*[1][self::mmlx:combination]]"
                      priority="100">
        <mrow>
```

```
              <mmultiscripts>
               <mi fontweight="bold">C</mi>
               <xsl:apply-templates select = "*[2]" mode = "semantics"/>
               <xsl:apply-templates select = "*[3]" mode = "semantics"/>
              </mmultiscripts>
             </mrow>
             </xsl:template>
           </presentation>
         </choice>

         <!-- ************ COMBINATION CHOICE 4 ************** -->
         <choice>
           <image src = "comb4.gif"/>
         <keyvalue> 4 </keyvalue>
           <presentation>
             <xsl:template match = "mml:apply[*[1][self::mmlx:combination]]"
                           priority="100">
              <mrow>
               <mmultiscripts>
                 <mi fontweight="bold">C</mi>
                 <none/>
                 <xsl:apply-templates select = "*[3]" mode = "semantics"/>
               <mprescripts/>
              <xsl:apply-templates select = "*[2]" mode = "semantics"/>
              <none/>
              </mmultiscripts>
             </mrow>
             </xsl:template>
           </presentation>
         </choice>
       </choicelist>
     </item>
   </itemlist>
 </catalog>



<!-- ************* Intervals ************ -->
  <catalog>
    <name>Intervals</name>
    <itemlist>

      <!-- ************* OPEN INTERVAL ************* -->
      <item>
        <title> Open Interval </title>
        <content>
          <!-- ???? →
        </content>
        <keyword> OPEN INTERVAL </keyword>
        <choicelist>

          <!-- ************* OPEN INTERVAL CHOICE 1 ************* -->
          <choice>
            <image src = "openinterval1.gif"/>
          <keyvalue> 1 </keyvalue>
            <presentation>
```

```
    <xsl:template match = "mml:interval" priority="100">
      <xsl:choose>
        <xsl:when test="@closure='closed'">
          <mfenced open="[" close="]" separators=",">
            <xsl:apply-templates select="*" mode = "semantics"/>
          </mfenced>
        </xsl:when>
        <xsl:when test="@closure='open'">
          <mfenced open="(" close=")" separators=",">
            <xsl:apply-templates select="*" mode = "semantics"/>
          </mfenced>
        </xsl:when>
        <xsl:when test="@closure='open-closed'">
          <mfenced open="(" close="]" separators=",">
            <xsl:apply-templates select="*" mode = "semantics"/>
          </mfenced>
        </xsl:when>
        <xsl:when test="@closure='closed-open'">
          <mfenced open="[" close=")" separators=",">
            <xsl:apply-templates select="*" mode = "semantics"/>
          </mfenced>
        </xsl:when>
        <xsl:otherwise>
          <mfenced open="[" close="]" separators=",">
            <xsl:apply-templates select="*" mode = "semantics"/>
          </mfenced>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:template>
  </presentation>
</choice>

<!-- ************* OPEN INTERVAL CHOICE 2 ************* -->
<choice>
  <image src = "openinterval2.gif"/>
<keyvalue> 2 </keyvalue>
  <presentation>
    <xsl:template match = "mml:interval">
      <xsl:choose>
        <xsl:when test="@closure='closed'">
          <mfenced open="[" close="]" separators=",">
            <xsl:apply-templates select="*" mode = "semantics"/>
          </mfenced>
        </xsl:when>
        <xsl:when test="@closure='open'">
          <mfenced open="]" close="[" separators=",">
            <xsl:apply-templates select="*" mode = "semantics"/>
          </mfenced>
        </xsl:when>
        <xsl:when test="@closure='open-closed'">
          <mfenced open="]" close="]" separators=",">
            <xsl:apply-templates select="*" mode = "semantics"/>
          </mfenced>
        </xsl:when>
        <xsl:when test="@closure='closed-open'">
          <mfenced open="[" close="[" separators=",">
            <xsl:apply-templates select="*" mode = "semantics"/>
```

```
                    </mfenced>
                  </xsl:when>
                  <xsl:otherwise>
                    <mfenced open="[" close="]" separators=",">
                      <xsl:apply-templates select="*" mode = "semantics"/>
                    </mfenced>
                  </xsl:otherwise>
                </xsl:choose>
              </xsl:template>
            </presentation>
          </choice>
        </choicelist>
      </item>
    </itemlist>
  </catalog>


<!-- *************** CALCULUS **************  -->
  <catalog>
    <name>Calculus</name>
    <itemlist>

    <!-- ***************** DERIVATIVE ************************** -->
      <item>
        <title> Derivative  </title>
        <content>
          <!-- ???? →
        </content>
        <keyword> DERIVATIVE </keyword>
        <choicelist>

          <!-- ***************** DERIVATIVE CHOICE 1 ********* -->
          <choice>
            <image src = "derivative1.gif"/>
            <keyvalue> 1 </keyvalue>
            <presentation>
              <xsl:template match = "mml:apply[mml:diff[1]]" priority="100">
                <mrow>
                  <msup>
                    <mrow>
                      <mo>(</mo>
                      <xsl:apply-templates select = "*[3]" mode = "semantics"/>
                      <mo>)</mo>
                    </mrow>
                    <xsl:if test="*[2]=mml:bvar and mml:bvar[*[2]=mml:degree]">
                      <mo>
                        <xsl:call-template name="prime">
                          <xsl:with-param name="n">
                            <xsl:value-of select="mml:bvar/*[2]/*"/>
                          </xsl:with-param>
                        </xsl:call-template>
                      </mo>
                    </xsl:if>

                  <!-- ********* degree = 1 ********** -->
                  <xsl:if test="*[2]=mml:bvar and
                                not(mml:bvar[*[2]=mml:degree])">
```

```
          <mo>,</mo>
        </xsl:if>
      </msup>
      </mrow>
    </xsl:template>

    <!-- create the string "''..'" (n "'"s) -->
    <xsl:template name="prime">
      <xsl:param name="n" select="1"/>
      <xsl:if test="$n > 0">
        <xsl:variable name="m" select="$n - 1"/>
        ,
        <xsl:if test="$m > 0">
          <xsl:call-template name="prime">
            <xsl:with-param name="n" select="$m"/>
          </xsl:call-template>
        </xsl:if>
      </xsl:if>
    </xsl:template>
  </presentation>
</choice>



<!-- *************** DERIVATIVE CHOICE 2 ********* -->
<choice>
  <image src = "derivative2.gif"/>
  <keyvalue> 2 </keyvalue>
  <presentation>
    <xsl:template match = "mml:apply[mml:diff[1]]" priority="100">
      <mrow>
        <xsl:if test="*[2]=mml:bvar and mml:bvar[*[2]=mml:degree]">
          <mfrac>
            <msup>
              <mo>d</mo>
              <xsl:apply-templates select = "mml:bvar/mml:degree"
                                   mode = "semantics"/>
            </msup>
            <mrow>
              <mo>d</mo>
              <msup>
                <xsl:apply-templates select = "mml:bvar/*[1]"
                                     mode = "semantics"/>
                <xsl:apply-templates select = "mml:bvar/mml:degree"
                                     mode = "semantics"/>
              </msup>
            </mrow>
          </mfrac>
        </xsl:if>
        <xsl:if test="*[2]=mml:bvar and
                      not(mml:bvar[*[2]=mml:degree])">
          <mfrac>
            <mo>d</mo>
            <mrow>
              <mo>d</mo>
              <xsl:apply-templates select = "mml:bvar/*[1]"
                                   mode = "semantics"/>
```

```
            </mrow>
          </mfrac>
        </xsl:if>
        <mo>(</mo>
          <xsl:apply-templates select = "*[3]" mode = "semantics"/>
        <mo>)</mo>
      </mrow>
    </xsl:template>
  </presentation>
</choice>

<!-- *************** DERIVATIVE CHOICE 3 ********* -->
<choice>
  <image src = "derivative3.gif"/>
  <keyvalue> 3 </keyvalue>
  <presentation>
    <xsl:template match = "mml:apply[mml:diff[1]]" priority="100">
      <mrow>
        <msub>
          <mo>D</mo>
        <xsl:if test="*[2]=mml:bvar and mml:bvar[*[2]=mml:degree]">
          <mo>
            <xsl:call-template name="degrees">
              <xsl:with-param name="what">
                 <xsl:value-of select="mml:bvar/*[1]"/>
              </xsl:with-param>
              <xsl:with-param name="n">
                 <xsl:value-of select="mml:bvar/*[2]/*"/>
              </xsl:with-param>
            </xsl:call-template>
          </mo>
        </xsl:if>

        <!-- ********* degree = 1 ********** -->
        <xsl:if test="*[2]=mml:bvar and
                      not(mml:bvar[*[2]=mml:degree])">
            <xsl:apply-templates select = "mml:bvar/*[1]"
                                 mode = "semantics"/>
        </xsl:if>
      </msub>
      <mrow>
        <mo>(</mo>
        <xsl:apply-templates select = "*[3]" mode = "semantics"/>
        <mo>)</mo>
      </mrow>
    </mrow>
  </xsl:template>

    <!-- create the string "xx..x" (n "what"s) -->
    <xsl:template name="degrees">
      <xsl:param name="what" select="x"/>
      <xsl:param name="n" select="1"/>
      <xsl:if test="$n > 0">
        <xsl:variable name="m" select="$n - 1"/>
        <xsl:value-of select="$what"/>
        <xsl:if test="$m > 0">
          <xsl:call-template name="degrees">
```

```
            <xsl:with-param name="what" select ="$what"/>
            <xsl:with-param name="n" select="$m"/>
          </xsl:call-template>
        </xsl:if>
      </xsl:if>
    </xsl:template>
  </presentation>
</choice>

<!-- ***************** DERIVATIVE CHOICE 4 ********* -->
<choice>
  <image src = "derivative4.gif"/>
  <keyvalue> 4 </keyvalue>
  <presentation>
    <xsl:template match = "mml:apply[mml:diff[1]]" priority="100">
      <mrow>
        <msub>
          <mrow>
            <mo>(</mo>
            <xsl:apply-templates select = "*[3]" mode = "semantics"/>
            <mo>)</mo>
          </mrow>
          <xsl:if test="*[2]=mml:bvar and mml:bvar[*[2]=mml:degree]">
            <mo>
              <xsl:call-template name="degrees">
                <xsl:with-param name="what">
                   <xsl:value-of select="mml:bvar/*[1]"/>
                </xsl:with-param>
                <xsl:with-param name="n">
                  <xsl:value-of select="mml:bvar/*[2]/*"/>
                </xsl:with-param>
              </xsl:call-template>
            </mo>
          </xsl:if>

        <!-- ********* degree = 1 ********** -->
        <xsl:if test="*[2]=mml:bvar and
                      not(mml:bvar[*[2]=mml:degree])">
          <xsl:apply-templates select = "mml:bvar/*[1]"
                               mode = "semantics"/>
        </xsl:if>
      </msub>
      </mrow>
    </xsl:template>

    <!-- create the string "xx..x" (n "what"s) -->
    <xsl:template name="degrees">
      <xsl:param name="what" select="x"/>
      <xsl:param name="n" select="1"/>
      <xsl:if test="$n > 0">
        <xsl:variable name="m" select="$n - 1"/>
        <xsl:value-of select="$what"/>
        <xsl:if test="$m > 0">
          <xsl:call-template name="degrees">
            <xsl:with-param name="what" select ="$what"/>
            <xsl:with-param name="n" select="$m"/>
          </xsl:call-template>
```

```
          </xsl:if>
        </xsl:if>
      </xsl:template>
    </presentation>
  </choice>


  <!-- *************** DERIVATIVE CHOICE 5 ********* -->
  <choice>
    <image src = "derivative5.gif"/>
    <keyvalue> 5 </keyvalue>
    <presentation>
      <xsl:template match = "mml:apply[mml:diff[1]]" priority="100">
        <mrow>
          <msup>
            <mrow>
              <mo>(</mo>
              <xsl:apply-templates select = "*[3]" mode = "semantics"/>
              <mo>)</mo>
            </mrow>
            <xsl:if test="*[2]=mml:bvar and mml:bvar[*[2]=mml:degree]">
              <mo>
                <xsl:call-template name="prime">
                  <xsl:with-param name="n">
                    <xsl:value-of select="mml:bvar/*[2]/*"/>
                  </xsl:with-param>
                </xsl:call-template>
              </mo>
            </xsl:if>

          <!-- ********* degree = 1 ********** -->
          <xsl:if test="*[2]=mml:bvar and
                        not(mml:bvar[*[2]=mml:degree])">
            <mo>.</mo>
          </xsl:if>
        </msup>
        </mrow>
      </xsl:template>

      <!-- create the string "...'" (n "."s) -->
      <xsl:template name="prime">
        <xsl:param name="n" select="1"/>
        <xsl:if test="$n > 0">
          <xsl:variable name="m" select="$n - 1"/>
          .
          <xsl:if test="$m > 0">
            <xsl:call-template name="prime">
              <xsl:with-param name="n" select="$m"/>
            </xsl:call-template>
          </xsl:if>
        </xsl:if>
      </xsl:template>
    </presentation>
  </choice>
 </choicelist>
</item>
```

```
<!-- **********  PARTIAL DERIVATIVE ************  -->
<item>
  <title>Partial derivative </title>
  <content>
    <!-- ???? →
  </content>
  <keyword> PARTIAL DERIVATIVE</keyword>
  <choicelist>

    <!-- **********  PARTIAL DERIVATIVE CHOICE 1 ************  -->
    <choice>
      <image src = "partiald1.gif"/>
      <keyvalue> 1 </keyvalue>
      <presentation>
        <xsl:template match = "mml:apply[mml:partialdiff[1]]">
          <mrow>
            <xsl:for-each select = "mml:bvar">
              <xsl:if test="*[last()]=mml:degree">
                <mfrac>
                  <msup>
                     <mo><mchar name= "PartialD"/> </mo>
                     <xsl:apply-templates select = "mml:degree"
                                          mode = "semantics"/>
                  </msup>
                  <mrow>
                    <mo><mchar name= "PartialD"/> </mo>
                    <msup>
                      <xsl:apply-templates select = "*[1]"
                                           mode = "semantics"/>
                      <xsl:apply-templates select = "mml:degree"
                                           mode = "semantics"/>
                    </msup>
                  </mrow>
                </mfrac>
              </xsl:if>
              <xsl:if test="not(*[last()]=mml:degree)">
                <mfrac>
                  <mo><mchar name= "PartialD"/></mo>
                  <mrow>
                    <mo><mchar name= "PartialD"/> </mo>
                    <xsl:apply-templates select = "*[1]"
                                         mode = "semantics"/>
                  </mrow>
                </mfrac>
              </xsl:if>
            </xsl:for-each>
            <mo>(</mo>
            <xsl:apply-templates select = "*[last()]"
                                 mode = "semantics"/>
            <mo>)</mo>
          </mrow>
        </xsl:template>
      </presentation>
    </choice>

    <!-- **********  PARTIAL DERIVATIVE CHOICE 2 ************  -->
    <choice>
```

```
  <image src = "partiald2.gif"/>
  <keyvalue> 2 </keyvalue>
  <presentation>
    <xsl:template match = "mml:apply[mml:partialdiff[1]]">
      <mrow>
        <msub>
          <mo><mchar name= "PartialD"/> </mo>
          <mo>
            <xsl:for-each select = "mml:bvar">
              <xsl:if test="*[last()]=mml:degree">
                <xsl:call-template name="degrees">
                  <xsl:with-param name="what">
                     <xsl:value-of select="*[1]"/>
                  </xsl:with-param>
                  <xsl:with-param name="n">
                     <xsl:value-of select="*[2]/*"/>
                  </xsl:with-param>
                </xsl:call-template>
              </xsl:if>

              <!-- ********* degree = 1 ********** -->
              <xsl:if test="not(*[last()]=mml:degree)">

                <xsl:apply-templates select = "*[1]"
                                     mode = "semantics"/>
              </xsl:if>
            </xsl:for-each>
          </mo>
        </msub>
        <mo>(</mo>
        <xsl:apply-templates select = "*[last()]"
                             mode = "semantics"/>
        <mo>)</mo>
      </mrow>
    </xsl:template>

    <!-- create the string "xx..x" (n "what"s) -->
    <xsl:template name="degrees">
      <xsl:param name="what" select="x"/>
      <xsl:param name="n" select="1"/>
      <xsl:if test="$n > 0">
        <xsl:variable name="m" select="$n - 1"/>
        <xsl:value-of select="$what"/>
        <xsl:if test="$m > 0">
          <xsl:call-template name="degrees">
            <xsl:with-param name="what" select ="$what"/>
            <xsl:with-param name="n" select="$m"/>
          </xsl:call-template>
        </xsl:if>
      </xsl:if>
    </xsl:template>

  </presentation>
</choice>

<!-- **********  PARTIAL DERIVATIVE CHOICE 3 ***********  -->
<choice>
```

```
  <image src = "partiald3.gif"/>
  <keyvalue> 3 </keyvalue>
  <presentation>
    <xsl:template match = "mml:apply[mml:partialdiff[1]]">
      <mrow>
        <msub>
          <mo>D</mo>
          <mo>
            <xsl:for-each select = "mml:bvar">
              <xsl:if test="*[last()]=mml:degree">
                <xsl:call-template name="degrees">
                  <xsl:with-param name="what">
                     <xsl:value-of select="*[1]"/>
                  </xsl:with-param>
                  <xsl:with-param name="n">
                     <xsl:value-of select="*[2]/*"/>
                  </xsl:with-param>
                </xsl:call-template>
              </xsl:if>

              <!-- ********* degree = 1 ********** -->
              <xsl:if test="not(*[last()]=mml:degree)">

                <xsl:apply-templates select = "*[1]"
                                     mode = "semantics"/>
              </xsl:if>

            </xsl:for-each>
          </mo>
        </msub>
        <mo>(</mo>
        <xsl:apply-templates select = "*[last()]"
                             mode = "semantics"/>
        <mo>)</mo>
      </mrow>
    </xsl:template>

    <!-- create the string "xx..x" (n "what"s) -->
    <xsl:template name="degrees">
      <xsl:param name="what" select="x"/>
      <xsl:param name="n" select="1"/>
      <xsl:if test="$n > 0">
        <xsl:variable name="m" select="$n - 1"/>
        <xsl:value-of select="$what"/>
        <xsl:if test="$m > 0">
          <xsl:call-template name="degrees">
            <xsl:with-param name="what" select ="$what"/>
            <xsl:with-param name="n" select="$m"/>
          </xsl:call-template>
        </xsl:if>
      </xsl:if>
    </xsl:template>
  </presentation>
</choice>

<!-- **********  PARTIAL DERIVATIVE CHOICE 4 ***********  -->
<choice>
```

```
            <image src = "partiald4.gif"/>
            <keyvalue> 4 </keyvalue>
            <presentation>
              <xsl:template match = "mml:apply[mml:partialdiff[1]]">
                <mrow>
                  <msub>
                    <mo><mchar name="Del"/> </mo>  <!-- bigtriangledown  Delta -
->
                    <mo>
                      <xsl:for-each select = "mml:bvar">
                        <xsl:if test="*[last()]=mml:degree">
                          <xsl:call-template name="degrees">
                            <xsl:with-param name="what">
                               <xsl:value-of select="*[1]"/>
                            </xsl:with-param>
                            <xsl:with-param name="n">
                               <xsl:value-of select="*[2]/*"/>
                            </xsl:with-param>
                          </xsl:call-template>
                        </xsl:if>

                        <!-- ********* degree = 1 ********** -->
                        <xsl:if test="not(*[last()]=mml:degree)">

                          <xsl:apply-templates select = "*[1]"
                                               mode = "semantics"/>
                        </xsl:if>
                      </xsl:for-each>
                    </mo>
                  </msub>
                  <mo>(</mo>
                  <xsl:apply-templates select = "*[last()]"
                                       mode = "semantics"/>
                  <mo>)</mo>
                </mrow>
              </xsl:template>

              <!-- create the string "xx..x" (n "what"s) -->
              <xsl:template name="degrees">
                <xsl:param name="what" select="x"/>
                <xsl:param name="n" select="1"/>
                <xsl:if test="$n > 0">
                  <xsl:variable name="m" select="$n - 1"/>
                  <xsl:value-of select="$what"/>
                  <xsl:if test="$m > 0">
                    <xsl:call-template name="degrees">
                      <xsl:with-param name="what" select ="$what"/>
                      <xsl:with-param name="n" select="$m"/>
                    </xsl:call-template>
                  </xsl:if>
                </xsl:if>
              </xsl:template>
            </presentation>
          </choice>
          <choice>
            <image src = "partiald5.gif"/>
            <keyvalue> 5 </keyvalue>
```

```
          <presentation>
            <xsl:template match = "mml:apply[mml:partialdiff[1]]">
              <mrow>
                <msub>
                  <mrow>
                    <mo>(</mo>
                    <xsl:apply-templates select = "*[last()]"
                                         mode = "semantics"/>
                    <mo>)</mo>
                  </mrow>
                  <mo>
                    <xsl:for-each select = "mml:bvar">
                      <xsl:if test="*[last()]=mml:degree">
                        <xsl:call-template name="degrees">
                          <xsl:with-param name="what">
                             <xsl:value-of select="*[1]"/>
                          </xsl:with-param>
                          <xsl:with-param name="n">
                             <xsl:value-of select="*[2]/*"/>
                          </xsl:with-param>
                        </xsl:call-template>
                      </xsl:if>

                      <!-- ********* degree = 1 ********** -->
                      <xsl:if test="not(*[last()]=mml:degree)">
                        <xsl:apply-templates select = "*[1]"
                                             mode = "semantics"/>
                      </xsl:if>
                    </xsl:for-each>
                  </mo>
                </msub>
              </mrow>
            </xsl:template>

            <!-- create the string "xx..x" (n "what"s) -->
            <xsl:template name="degrees">
              <xsl:param name="what" select="x"/>
              <xsl:param name="n" select="1"/>
              <xsl:if test="$n > 0">
                <xsl:variable name="m" select="$n - 1"/>
                <xsl:value-of select="$what"/>
                <xsl:if test="$m > 0">
                  <xsl:call-template name="degrees">
                    <xsl:with-param name="what" select ="$what"/>
                    <xsl:with-param name="n" select="$m"/>
                  </xsl:call-template>
                </xsl:if>
              </xsl:if>
            </xsl:template>
          </presentation>
        </choice>
      </choicelist>
    </item>
  </itemlist>
</catalog>
```

```
<!-- *************LINEAR ALGEBRA ************************  -->
  <catalog>
    <name>Linear Algebra</name>
    <itemlist>

      <!-- ********* VECTOR ****************************  -->
      <item>
        <title> Vector </title>
        <content>
           <!-- ???? →
        </content>
        <keyword> VECTOR </keyword>
        <choicelist>

          <!-- ********* VECTOR CHOICE 1 *********************  -->
          <choice>
            <image src = "vect1.gif"/>
            <keyvalue> 1 </keyvalue>
            <presentation>
              <xsl:template match = "mml:ci[@type='vector']" priority="100">
                <mi fontweight="bold" fontstyle="normal">
                  <xsl:apply-templates mode = "semantics"/>
                </mi>
              </xsl:template>
            </presentation>
          </choice>

          <!-- ********* VECTOR CHOICE 2 *********************  -->
          <choice>
            <image src = "vect2.gif"/>
            <keyvalue> 2 </keyvalue>
            <presentation>
              <xsl:template match = "mml:ci[@type='vector']" priority="100">
                <mi fontstyle="italic">
                  <xsl:apply-templates mode = "semantics"/>
                </mi>
              </xsl:template>
            </presentation>
          </choice>

          <!-- ********* VECTOR CHOICE 3 *********************  -->
          <choice>
            <image src = "vect3.gif"/>
            <keyvalue> 3 </keyvalue>
            <presentation>
              <xsl:template match = "mml:ci[@type='vector']" priority="100">
                <mrow>
                  <mover accent="true">
                    <mi fontstyle="italic">
                      <xsl:apply-templates mode = "semantics"/>
                    </mi>
                    <mo> <mchar name="RightArrow"/> </mo>
                  </mover>
                </mrow>
              </xsl:template>
            </presentation>
          </choice>
```

```
    <!-- ********* VECTOR CHOICE 4 *********************  -->
    <choice>
      <image src = "vect4.gif"/>
      <keyvalue> 4 </keyvalue>
      <presentation>
      <xsl:template match = "mml:ci[@type='vector']" priority="100">
          <mrow>
            <msub>
              <mi fontstyle="italic">
                <xsl:apply-templates mode = "semantics"/>
              </mi>
              <mi fontstyle="italic">i </mi>
            </msub>
          </mrow>
        </xsl:template>
      </presentation>
    </choice>
  </choicelist>
</item>

<!-- *********  MATRIX ****************************** -->
<item>
  <title>Matrix </title>
  <content>
    <!-- ???? →
  </content>
  <keyword> MATRIX</keyword>
  <choicelist>

    <!-- *********  MATRIX CHOICE 1 ********************** -->
    <choice>
      <image src = "matrix1.gif"/>
      <keyvalue> 1 </keyvalue>
      <presentation>
        <xsl:template match = "mml:ci[@type='matrix']" priority="100">
          <mi fontweight="bold" fontstyle="normal">
            <xsl:apply-templates mode = "semantics"/>
          </mi>
        </xsl:template>
      </presentation>
    </choice>

    <!-- *********  MATRIX CHOICE 2 ********************** -->
    <choice>
      <image src = "matrix2.gif"/>
      <keyvalue> 2 </keyvalue>
      <presentation>
        <xsl:template match = "mml:ci[@type='matrix']" priority="100">
          <mi fontstyle="italic">
            <xsl:apply-templates mode = "semantics"/>
          </mi>
        </xsl:template>
      </presentation>
    </choice>

    <!-- *********  MATRIX CHOICE 3 ********************** -->
```

```
      <choice>
        <image src = "matrix3.gif"/>
        <keyvalue> 3 </keyvalue>
        <presentation>
          <xsl:template match = "mml:ci[@type='matrix']" priority="100">
            <mrow>
              <mover accent="true">
                <mi fontstyle="italic">
                  <xsl:apply-templates mode = "semantics"/>
                </mi>
                <mo> <mchar name="LeftRightArrow"/> </mo>
              </mover>
            </mrow>
          </xsl:template>
        </presentation>
      </choice>

      <!-- *********  MATRIX CHOICE 4 ********************** -->
      <choice>
        <image src = "matrix4.gif"/>
        <keyvalue> 4 </keyvalue>
        <presentation>
        <xsl:template match = "mml:ci[@type='matrix']" priority="100">
            <mrow>
              <msub>
                <mi fontstyle="italic">
                  <xsl:apply-templates mode = "semantics"/>
                </mi>
                <mi fontstyle="italic">ij </mi>
              </msub>
            </mrow>
          </xsl:template>
        </presentation>
      </choice>

      <!-- *********  MATRIX CHOICE 1 ********************** -->
      <choice>
        <image src = "matrix5.gif"/>
        <keyvalue> 5 </keyvalue>
        <presentation>
        <xsl:template match = "mml:ci[@type='matrix']" priority="100">
            <mrow>
              <mo>[</mo>
              <msub>
                <mi fontstyle="italic">
                  <xsl:apply-templates mode = "semantics"/>
                </mi>
                <mi fontstyle="italic">ij </mi>
              </msub>
              <mo>]</mo>
            </mrow>
          </xsl:template>
        </presentation>
      </choice>
    </choicelist>
  </item>
</itemlist>
```

```
  </catalog>
</mnotations>
```

Appendix B – Source Code

B.1 Catalog.java

```
package Project.MathTools;

import java.util.LinkedList;


/**
 * A math catalog has a list of items
 */
public class Catalog implements Comparable {
    private String name;
    private SortedList itemList;

    /**
     * Implements interface Comparable
     * @param: obj is an object to which this Catalog object compares
     * @return:the value 0if the name of obj is lexicographically equal
     * to the name of this catalog; a value less than 0 if the name of
     * obj is a string lexicographically greater than the name of this
     * catalog; and a value greater than 0 if the name of the obj is a
     * string lexicographically less than the name of this catalog
     */
    public int compareTo(Object obj){
        Catalog cat = (Catalog)obj;
        return (name.compareTo(cat.getName()));
    }


    /**
     *  Constructor: initializes a Catalog object
     *  @param: catalogName is the name of a catalog
     */
    public Catalog(String catalogName){
       name = catalogName;
       itemList = new SortedList();
    }


    /**
     *   Constructor: initializes a Catalog object
     *  @param: none
     */
    public Catalog(){
       name = null;
       itemList = new SortedList();
    }


    /**
     * returns the name of this catalog
     * param: none
     */
    public String getName(){
       return name;
    }
```

```
/**
 * returns the item list of this catalog
 * param: none
 */
public SortedList getItemList(){
   return itemList;
}


/**
 * Sets the name of this Catalog object to be catalogName
 * param: catalogName is the name of this catalog
 */
public void setName(String catalogName){
   name = catalogName;
}


/**
 * Adds an Item object item to this catalog's item list
 * @param: item is an Item object to be added
 */
public void addItem(Item item){
   itemList.insert(item);
}


/**
 *  Finds the item whose keyword contains the parameter subkeyword if
 *  any, otherwise null.
 *  @param: subkeyword is a String which might be a substring of an item's
 *          keyword in this catalog
 *  @return: the item if any, otherwise null
 */
public Item findItem(String subkeyword){
   for(int i=0; i<itemList.size(); i++){
       Item item = (Item)(itemList.get(i));
       String keyword = item.getKeyword();
       int dif = keyword.length() - subkeyword.length() ;
       for(int j=0; j<= dif; j++){
          if (keyword.substring(j).startsWith(subkeyword.toUpperCase())){
              return item;

          }

       }
   }
   return null;
}


/** Finds the index of the item whose keyword contains subkeyword as
 *  substring
 *  @param:  subkeyword is a String which might be a substring of an item's
 *          keyword in this catalog
 *  @return: the index of the item if any, otherwise -1.
public int indexOfItem(String subkeyword){
   for(int i=0; i<itemList.size(); i++){
       Item item = (Item)(itemList.get(i));
       String keyword = item.getKeyword();
       int dif = keyword.length() - subkeyword.length() ;
```

```
            for(int j=0; j<= dif; j++){
               if (keyword.substring(j).startsWith(subkeyword.toUpperCase())){
                   return i;

               }
            }
         }
         return -1;

      }


      /**
       *Overrides the toString method in Object
       *@return: a String containing the information of the catalog
       *@param: none
       */
      public String toString(){
         String ret;
         ret = "name: " + name +"\n";
         for (int i=0; i<itemList.size(); i++){
             Item item = (Item)(itemList.get(i));
             ret = ret +"\n" + item.toString();
         }
         ret = ret + "\n";
          return ret;

      }

}
```

B.2 CatalogPanel.java

```
package Project.MathTools;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

/**
 * Defines the common behaviors of RadioButtonCatalogPanel and
 * ComboBoxCatalogPanel classes
 */
public abstract class CatalogPanel extends JPanel{
     protected Catalog catalog;

    /**
     * creates and adds all the item panels to this panel
     */
    protected abstract void addItemPanels();
}
```

B.3 Choice.java

```
package Project.MathTools;

import org.w3c.dom.*;
import org.apache.xerces.parsers.DOMParser;

/**
 * A math notation choice
 */
public class Choice {
    private String image;
    private String keyvalue;
    private Element presentation;

    /**
     * Constructor: creates a new Choice Object
     * @param: img is the image file name containing the notation image
     * @param: value is the keyvalue of this choice
     */
    public Choice(String img, String value, Element e){
      image = img;
      keyvalue = value;
      presentation = e;
    }


    /**
     * Returns the image file name of this Choice
     * @param: none
     */
    public String getImage(){
      return image;
    }


    /**
     * Returns the keyvalue of this Choice
     * @param: none
     */
    public String getKeyvalue(){
      return keyvalue;
    }

    /**
     * Returns the presentation Element of this Choice
     * @param: none
     */
    public Element getPresentation(){
      return presentation;
    }

    /**
     * Sets the notation image file name
     * @param: none
     */
```

```
    public void setImage(String img){
      image = img;
    }


    /**
     * Sets the keyvalue
     * @param: value is the keyvalue of this Choice
     */
    public void setKeyvalue(String value){
      keyvalue = value;
    }

    /**
     * Sets the presentation to be the Element e
     * @param: e is the presentation Element
     */
    public void setPresentation(Element e){
      presentation = e;
    }

    /**
     *Overrides the toString method in Object
     *@return: a String of the information of the Choice
     *@param: none
     */
    public String toString(){
      return ("image src: " +image + "\t" +"keyvalue: " + keyvalue
+"\n");
    }
}
```

B.4 ComboBoxCatalogPanel.java

```
package Project.MathTools;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import java.util.*;

/**
 *  A ComboBoxCatalogPanel object contains a set of ComboBoxItemPanel objects
 *  and shows all the items and their choices of a catalog. All choices are
 *  orgnized in a combo box
 */
public class ComboBoxCatalogPanel extends CatalogPanel{
    //private Catalog catalog;

    /**
     *  Constructor: initializes a panel for a catalog. The panel uses GridBag
     *               layout.
     *  @param: cat is a Catalog object corresponding this panel
     */
    public ComboBoxCatalogPanel(Catalog cat){
       catalog = cat;
       addItemPanels();

    }

    /*
     * Creates and adds all ItemPanels to this Catalog panel.
     * @return: none
     * @parm: none
     */
    protected void addItemPanels(){
       LinkedList list = catalog.getItemList();
       int size = list.size();

        GridBagLayout gridbag = new GridBagLayout();
        GridBagConstraints c = new GridBagConstraints();
        setLayout(gridbag);
       c.gridwidth = 1;
       c.anchor = GridBagConstraints.WEST;
       c.insets = new Insets(0, 10, 0, 10);

       Item item;
       for(int i=0; i<list.size(); i++){
           item = (Item)(list.get(i));
           JPanel panel = new ComboBoxItemPanel(item);
           //panel.setMaximumSize( new Dimension(600, 60));
           //panel.setPreferredSize(new Dimension(600, 60));
           //panel.setBackground(Color.lightGray);
            c.weightx = 0.9;

            c.gridx = 0;
            c.gridy = i;
            gridbag.setConstraints(panel, c);
             add(panel);
       }
    }
}
```

B.5 ComboBoxItemPanel.java

```
package Project.MathTools;

import org.w3c.dom.*;
import org.apache.xerces.parsers.DOMParser;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import java.util.*;

/**
 * A ComboBoxItemPanel contains a text field and a combo box which
contains
 * a list of choices. It corresponds a math item
 */
public class ComboBoxItemPanel extends ItemPanel
    implements ActionListener{
    //private Item item;
    private JComboBox box;
    /**
     * Default constructor
     */
    public ComboBoxItemPanel(){}

    /**
     *  Constructor: create a ComboBoxItemPanel object
     *  @param: it is a math item
     */
    public ComboBoxItemPanel(Item it){

      item = it;
        setLayout( new GridLayout(1, 2, 1, 0));
      //setMaximumSize(new Dimension(600, 60));
      setPreferredSize(new Dimension(600, 60));
      //addItemLabel();

      addItemText();

      addChoiceBox();
      //box.setForeground(getBackground());

    }

    /**
     *  Creates the text field and adds it to this panel
     *  @param: none
     *  @return: none
     */
    protected void addItemText(){
      String name = item.getKeyword();
      textField  = new JTextField(name );
      textField.setEditable(false);
      textField.setBorder(null);
      add(textField);
```

```
    }


    /**
     *  Creates the combo box and adds it to this panel
     *  @param: none
     *  @return: none
     */
    protected void addChoiceBox(){
      LinkedList list = item.getChoicelist();
      box = new JComboBox();
      box.setEditable(false);
      Choice choice;
      String imagefile;
      ImageIcon icon;

      String key = item.getSelection();
      int selectedIndex = 0;
      for (int i = 0; i < list.size(); i++){
          choice = (Choice)(list.get(i));
          if(key.equals(choice.getKeyvalue()))
            selectedIndex = i;
          imagefile = choice.getImage();
          icon = new ImageIcon("../images/" + imagefile);
          box.addItem(icon);
      }

      box.setSelectedIndex(selectedIndex);

      box.addActionListener(this);
      add(box);

    }

    /**
     *  returns the text field
     *  @param: none
     *  @return: the text field
     */
    public JTextField getTextField(){
      return textField;
    }

    /**
     *  Implements the interface ActionListener
     */
    public void actionPerformed(ActionEvent evt){
      JComboBox source = (JComboBox)(evt.getSource());
      int index = source.getSelectedIndex();
      Choice choice = (Choice)(item.getChoicelist().get(index));
      String keyvalue = choice.getKeyvalue();
      //Element e = choice.getPresentation();
      item.setSelection(keyvalue);
      //item.setSelectedPresentation(e);
      MathNotationsFrame.isSaved = false;
    }
}
```

B.6 Comparable.java

```
package Project.MathTools;

public interface Comparable{
  /**
   * compares the object implementing Comparable to compareMe
   * @return an int <0 if compareMe is bigger than the object,
   *         0 if compareMe is indistinguishable from the object ,
   *         and an int >0 if compareMe is smaller than the object.
   */
  public int compareTo(Object compareMe);
}
```

B.7 DataReader.java

```
package Project.MathTools;

import org.apache.xerces.parsers.DOMParser;
import org.w3c.dom.*;
import org.xml.sax.SAXException;
import java.io.IOException;
import org.apache.xml.serialize.*;
import java.util.*;
import javax.swing.JOptionPane;

/**
 *  A DataReader object extracts the data from a parsed XML document
 *  and contructs a list of Catalog objects
 */
public class DataReader{
    private Document document;
    private SortedList catalogList;


    /**
     *  Constructor: creates a DataReader object
     *  @param: doc is the XML Document from which to extract data.
     */
    public DataReader(Document doc){
       document = doc;
       catalogList = new SortedList();

        Element e = document.getDocumentElement(); //mnotations
        NodeList catList = e.getElementsByTagName("catalog");

        int numberOfCats = catList.getLength(); // number of catalogs
              Node node;

       for (int i=0; i<numberOfCats; i++){
           node = catList.item(i);    //catalog
           Catalog cat = readCatalog(node);
           if (cat != null)
              catalogList.insert( cat );
       }
    }


    /**
     *  Gets the list of Catalogs
     *  @param: none
     *  @return: A LinkedList which is a list of Catalogs
     */
    public SortedList getCatalogs(){
       return catalogList;

    }




    /**
     *  Reads a Catalog node in the XML document and constructs
     *  a Catalog onject
     *  @param: node is a Catalog Node in the XML tree
```

```
 *  @return: A Catalog object associated with this node
 */
public Catalog readCatalog(Node node){
   if(node == null)
       return null;

   String tag = ((Element)node).getTagName();

   if(!tag.equals("catalog")){
       JOptionPane.showMessageDialog
          (null, "tag name is not 'catalog' !",
           "Warning", JOptionPane.WARNING_MESSAGE);

       //System.out.println("tag name is not 'catalog'");
       return null;
   }

   NodeList  nameNodeList =
       ((Element)node).getElementsByTagName("name");
   int numberOfNames = nameNodeList.getLength();
   if( numberOfNames != 1){
       if(numberOfNames < 1)
          JOptionPane.showMessageDialog
              (null, "No name element in a catalog !",
               "Warning", JOptionPane.WARNING_MESSAGE);
          //System.out.println("no name element in a catalog");
       else if(numberOfNames > 1)
          JOptionPane.showMessageDialog
              (null, "Too many name elements in a catalog !",
               "Warning", JOptionPane.WARNING_MESSAGE);
          //System.out.println("too many name elements in a catalog");
       return null;
   }

   Node  nameNode = nameNodeList.item(0);
   String name = nameNode.getFirstChild().getNodeValue().trim();

   NodeList  itemlistNodeList =
       ((Element)node).getElementsByTagName("itemlist");
   int numberOfItemlists = itemlistNodeList.getLength();


   if( numberOfItemlists != 1){
       if(numberOfItemlists < 1)
          JOptionPane.showMessageDialog
              (null, "No itemlist  element in a catalog !",
               "Warning", JOptionPane.WARNING_MESSAGE);

          //System.out.println("no itemlist element in a catalog");
       else if(numberOfItemlists > 1)
          JOptionPane.showMessageDialog
              (null, "Too many itemlist elements in a catalog !",
               "Warning", JOptionPane.WARNING_MESSAGE);
          //System.out.println("too many itemlist elements in a catalog");
       return null;
   }

   Node  itemlistNode = itemlistNodeList.item(0);//itemlist node
   NodeList  itemNodeList =
       ((Element)itemlistNode).getElementsByTagName("item");
   int numberOfItems = itemNodeList.getLength();
```

```
   if(numberOfItems <= 0){
       JOptionPane.showMessageDialog
          (null, "No item element in a catalog " + name +" !",
           "Warning", JOptionPane.WARNING_MESSAGE);
       //System.out.println("No items in catalog " + name);
       return null;
   }

   Catalog catalog = new Catalog(name);
   for(int i=0; i<numberOfItems; i++){
       Node itemNode = itemNodeList.item(i);
       Item item = readItem(itemNode);
       if(item!=null)
          catalog.addItem(item);
   }

   if(catalog.getItemList().size() > 0)
       return catalog;
   else
       return null;
}


/**
 *  Reads a Item node in the XML document and constructs
 *  a Item object
 *  @param: node is a Item Node in the XML tree
 *  @return: A Catalog object associated with this node
 */
public Item readItem(Node node){
   if(node == null)
       return null;

   String tag = ((Element)node).getTagName();

   if(!tag.equals("item")){
       JOptionPane.showMessageDialog
          (null, "Tag name is not 'item' !",
           "Warning", JOptionPane.WARNING_MESSAGE);
       //System.out.println("tag name is not 'item'");
       return null;
   }

   String title = readTitle(node);

   if(title == null || title.equals(""))
       return null;

   String keyword = readKeyword(node);

   if(keyword == null || keyword.equals(""))
       return null;

   LinkedList choicelist = readChoiceList(node);

   if(choicelist == null){
       return null;
   }



   NodeList  contentNodeList =
```

```
        ((Element)node).getElementsByTagName("content");

    int numberOfCons = contentNodeList.getLength();

    if( numberOfCons != 1){
        if(numberOfCons < 1)
           JOptionPane.showMessageDialog
                (null, "No content element in a item !",
                 "Warning", JOptionPane.WARNING_MESSAGE);
           //System.out.println("no content element in a item");
        else if(numberOfCons > 1)
           JOptionPane.showMessageDialog
                (null, "too many content elements in a choice !",
                 "Warning", JOptionPane.WARNING_MESSAGE);

           //System.out.println("too many content elements in a choice");
        return null;
    }

    Node  conNode = contentNodeList.item(0);

    Element con = (Element)conNode;



    Item item = new Item(title, keyword, con, choicelist);
    return item;


}


/**
 *  Reads the title of an Item
 *  @param: node is the parent node of title
 *  @return: the title of this Item node
 */
public String readTitle(Node node){ //node is the parent node of title
    NodeList  titleNodeList =
        ((Element)node).getElementsByTagName("title");
     int numberOfTitles = titleNodeList.getLength();

    if( numberOfTitles != 1){
        if(numberOfTitles < 1)
           JOptionPane.showMessageDialog
                (null, "No title element in an item !",
                 "Warning", JOptionPane.WARNING_MESSAGE);
        //System.out.println("no title element in an item");

        else if(numberOfTitles > 1)
           JOptionPane.showMessageDialog
                (null, "too many title elements in a item !",
                 "Warning", JOptionPane.WARNING_MESSAGE);
           //System.out.println("too many title elements in a item");
        return null;
    }
    Node  titleNode = titleNodeList.item(0);
    String titleName = titleNode.getFirstChild().getNodeValue().trim();

    if(titleName.equals(""))
        return null;

    return titleName;
```

```
}


/**
 *  Reads the keyword of an Item
 *  @param: node is the parent node of title
 *  @return: the keyword of this Item node
 */
public String readKeyword(Node node){ //node is the parent node of keyword
   NodeList  keywordNodeList =
       ((Element)node).getElementsByTagName("keyword");
   int numberOfKeywords = keywordNodeList.getLength();

   if( numberOfKeywords != 1){
       if(numberOfKeywords < 1)
          JOptionPane.showMessageDialog
               (null, "No keyword element in an item !",
                "Warning", JOptionPane.WARNING_MESSAGE);
       //System.out.println("no keyword element in a item");
       else if(numberOfKeywords > 1)
          JOptionPane.showMessageDialog
               (null, "too many keyword elements in an item !",
                "Warning", JOptionPane.WARNING_MESSAGE);
       //System.out.println("too many keyword elements in a item");
       return null;
   }

   Node  keywordNode = keywordNodeList.item(0);
   String keyword = keywordNode.getFirstChild().getNodeValue().trim();

   if(keyword.equals(""))
      return null;

   return keyword;
}


/**
 *  Reads the Choice list  of an Item
 *  @param: node is the parent node of title
 *  @return: the choice list which is a LinkedList object of this Item node
 */
public LinkedList readChoiceList(Node node){
   //node is the parent node of choicelist
   NodeList  choicelistNodeList =
       ((Element)node).getElementsByTagName("choicelist");
   int numberOfChoicelists = choicelistNodeList.getLength();

   if( numberOfChoicelists != 1){
       if(numberOfChoicelists < 1)
          JOptionPane.showMessageDialog
               (null, "No choicelist element in an item !",
                "Warning", JOptionPane.WARNING_MESSAGE);
          //System.out.println("no choicelist element in an item");
       else if(numberOfChoicelists > 1)
          JOptionPane.showMessageDialog
               (null, "too many choicelist elements in a item !",
                "Warning", JOptionPane.WARNING_MESSAGE);
       //System.out.println("too many choicelist elements in a item");
       return null;
   }
```

```
   Node  choicelistNode = choicelistNodeList.item(0);
   NodeList choiceNodeList =
       ((Element)choicelistNode).getElementsByTagName("choice");
   LinkedList choices = new LinkedList();

   for(int j=0; j<choiceNodeList.getLength(); j++){
       Node choiceNode = choiceNodeList.item(j);
       Choice ch = readChoice(choiceNode);
       if (ch != null)
          choices.addLast(ch);

   }

   if (choices.size() <= 0){
       return null;
   }

   return choices;

}


/**
 *  Reads a choice of an Item
 *  @param: node is the Choice node in the XML tree
 *  @return: a Choice object associated with this node
 */
public Choice readChoice(Node node){
   if(node == null)
       return null;

   String tag = ((Element)node).getTagName();

   if(!tag.equals("choice")){
          JOptionPane.showMessageDialog
              (null, "Tag name is not 'choice' !",
               "Warning", JOptionPane.WARNING_MESSAGE);
          //System.out.println("tag name is not 'choice'");
       return null;
   }

   String image = readImage(node);

   String keyvalue = readKeyvalue(node);

   if(image==null || keyvalue == null ||
      image.equals("")||keyvalue.equals(""))
       return null;

   NodeList  presentationNodeList =
       ((Element)node).getElementsByTagName("presentation");

   int numberOfPres = presentationNodeList.getLength();

   if( numberOfPres != 1){
       if(numberOfPres < 1)
          JOptionPane.showMessageDialog
              (null, "No presentation element in a choice !",
               "Warning", JOptionPane.WARNING_MESSAGE);
       //System.out.println("no presentation element in a choice");
       else if(numberOfPres > 1)
          JOptionPane.showMessageDialog
              (null, "Too many presentation elements in a choice !",
```

```
                "Warning", JOptionPane.WARNING_MESSAGE);
        //System.out.println("too many presentation elements in a choice");
        return null;
    }

    Node  preNode = presentationNodeList.item(0);

    Element pre = (Element)preNode;
    return( new Choice(image, keyvalue, pre));

}



/**
 *  Reads the Image Node which is the child node of parameter node
 *  @param: node is the Choice node which contains a Image Child node
 *  @return: the file name of the image
 */
public String readImage(Node node){//node is the parent node of image
    NodeList  imageNodeList =
        ((Element)node).getElementsByTagName("image");
    int numberOfImages = imageNodeList.getLength();

    if( numberOfImages != 1){
        if(numberOfImages < 1)
           JOptionPane.showMessageDialog
               (null, "No image element in a choice !",
                "Warning", JOptionPane.WARNING_MESSAGE);
           //System.out.println("no image element in a choice");
        else if(numberOfImages > 1)
           JOptionPane.showMessageDialog
               (null, "Too many image elements in a choice !",
                "Warning", JOptionPane.WARNING_MESSAGE);
        //System.out.println("too many image elements in a choice");
        return null;
    }

    Node  imageNode = imageNodeList.item(0);

    String image = ((Element)imageNode).getAttribute("src").trim();

    if(image.equals(""))
       return null;

    return image;
}



/**
 *  Reads the Keyvalue Node which is the child node of parameter node
 *  @param: node is the Choice node which contains a keyvalue child node
 *  @return: the keyvalue of this Choice
 */
public String readKeyvalue(Node node){//node is the parent node of keyvalue
    NodeList  keyvalueNodeList =
        ((Element)node).getElementsByTagName("keyvalue");
    int numberOfKeyvalues = keyvalueNodeList.getLength();

    if( numberOfKeyvalues != 1){
        if(numberOfKeyvalues < 1)
           JOptionPane.showMessageDialog
```

```
                (null, "No keyvalue element in a choice !",
                 "Warning", JOptionPane.WARNING_MESSAGE);
            //System.out.println("no keyvalue element in a choice");
            else if(numberOfKeyvalues > 1)
               JOptionPane.showMessageDialog
                (null, "Too many keyvalue elements in a choice !",
                 "Warning", JOptionPane.WARNING_MESSAGE);
            //System.out.println("too many keyvalue elements in a choice");
            return null;
         }
         Node  keyvalueNode = keyvalueNodeList.item(0);
         String keyvalue = keyvalueNode.getFirstChild().getNodeValue().trim();

         if(keyvalue.equals(""))
            return null;

         return keyvalue;

      }
   }
```

B.8 FindFrame.java

```
package Project.MathTools;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

import java.util.*;

/**
 *  A FindFrame object is generated when the user clicking on the Find button
 *  on the Notation Selection tool interface. It lets the user to find
 *  keywords which has the input string as substring in order.
 */
public class FindFrame extends JFrame implements ActionListener{
    private JTextField         field;
    private String             previous = null;
    private MathNotationsFrame origin;
    private int                cat_index=0;
    private int                item_index=-1;
    private boolean            atEnd = false;


    /**
     *  Constructor: creates a FindFrame object
     *  @param: frame is a MathNotationsFrame generating this FindFrame
     */
    public FindFrame(MathNotationsFrame frame){

       super("Math Notations:   Find");
       setLocation(400, 300);
       setSize(200, 100);
       field = new JTextField("", 20);
       //field.requestDefaultFocus();
       origin = frame;

       JLabel question =
           new JLabel( new ImageIcon("images/question.gif"));

       JLabel prompt = new JLabel("    Enter Keyword: ");

       JPanel panel = new JPanel();
       JButton find = new JButton("Find");
       find.setActionCommand("Find");
       find.addActionListener(this);
       panel.add(find);

       JButton clear = new JButton("Clear");
       clear.setActionCommand("Clear");
       clear.addActionListener(this);
       panel.add(clear);

       JButton close = new JButton("Close");
       close.setActionCommand("Close");
       close.addActionListener(this);
       panel.add(close);

       Container content = getContentPane();
       content.add(prompt, "North");
       content.add(question, "West");
       content.add(field, "Center");
```

```
   content.add(panel, "South");

   //field.requestDefaultFocus();
   pack();
   setVisible(true);
   field.requestFocus();
}


/**
 *  Implements the interface ActionListener
 */
public void actionPerformed(ActionEvent evt){
   String command = evt.getActionCommand();
   if (command.equals("Close")){//click on the Close button
       origin.disposeFindFrame();
       dispose();
       return;
   }

   if(command.equals("Clear")){//click on the Clear button
       field.setText("");
        previous = null;
       cat_index = 0;
       item_index = -1;
       field.requestFocus();
       return;

   }

   if (command.equals("Find")){ //click on the Find Button
             //Color background = origin.getTop().getBackground();
       String text  =  field.getText();
       if (text != null && !text.equals("") ){
          /*JOptionPane.showMessageDialog
              (null, "Nothing to find!",
               "Warning", JOptionPane.WARNING_MESSAGE);
          return;
          */

          boolean sameInput = true;
          if(!text.equals(previous)){
              cat_index = 0;
              item_index = -1;
              sameInput = false;


          }
          Item item = find(text);

          if (item != null){
              JTabbedPane jtp = origin.getTabbedPane();
              jtp.setSelectedIndex(cat_index);
              JScrollPane scroll_pane =
                 (JScrollPane)(jtp.getSelectedComponent());
              JScrollBar bar =  scroll_pane.getVerticalScrollBar();
                 JViewport viewport = scroll_pane.getViewport();
                 JPanel catpanel = (JPanel) (viewport.getView());

                 Component[] ps = catpanel.getComponents();

                 ItemPanel itempanel = (ItemPanel)ps[item_index];
                 JTextField textField = itempanel.getTextField();
                 String keyword = textField.getText();
                 int begin = keyword.indexOf(text.toUpperCase());
```

```
                //System.out.println("Begin = " + begin);

                /*
                 for (int j=0; j<ps.length; j++){
                    ((ItemPanel)ps[j]).getTextField().getHighlighter().
                       removeAllHighlights();
                }
                */


                textField.select(begin, begin + text.length());

                //itempanel.setBackground(Color.magenta);
                int location = (int)(itempanel.getLocation().getY());

                if(location >= bar.getValue() + bar.getVisibleAmount()
                       || location < bar.getValue())
                    bar.setValue(location);

                origin.validate();
                return;
            }

            if (sameInput == false){ //keyword not found
                JOptionPane.showMessageDialog
                   (null, text + " not found!", "Warning",
                    JOptionPane.WARNING_MESSAGE);
            }

            else { //no more items
                JOptionPane.showMessageDialog
                   (null, "No more " + text, "Warning",
                    JOptionPane.WARNING_MESSAGE);
            }
            return;

        }

        if (text != null && text.equals("")){
           JOptionPane.showMessageDialog
               (null, "Nothing to find!",
                "Warning", JOptionPane.WARNING_MESSAGE);
           return;

        }

    }

}



/**
 *  Finds the Item object which has text as its Keyword's substring
 *  @param: text is a String which is a substring of some keywords
 *  @return: An Item object which has text as its Keyword's substring
 */
public Item find(String text){
   LinkedList catalogList = origin.getCatalogList();
   int size = catalogList.size();
   Catalog cat;
   for(int i=cat_index; i<size; i++){
       cat = (Catalog)(catalogList.get(i));
       LinkedList itemList = cat.getItemList();
       for (int j=item_index+1; j<itemList.size(); j++){
```

```
                Item item = (Item)(itemList.get(j));
                if(item.contains(text)){
                    item_index = j;
                    cat_index = i;
                    previous = text;
                    atEnd = false;
                    return item;
                }

            }
            item_index = -1;
        }

        atEnd = true;
        return null;
    }

}
```

B.9 Item.java

```
package Project.MathTools;

import org.w3c.dom.*;
import org.apache.xerces.parsers.DOMParser;

import java.util.LinkedList;

/**
 * A math item has a list of choices
 */
public class Item implements Comparable {
    private String title;
    private Element content;
    private String keyword;
    private LinkedList choicelist;
    private String selection;
    private Element selectedPresentation;
    /**
     * Implements interface Comparable
     * @param: obj is an object to which this Item object compares
     * @return: the value 0 if the keyword of obj is lexicographically equal
     * to the keyword of this item; a value less than 0 if the keyword of obj
     * is a string lexicographically greater than the keyword of this item;
     * and a value greater than 0 if the keyword of the obj is a string
     * lexicographically less than the keyword of this item
     */
    public int compareTo(Object obj){
        Item item = (Item)obj;
        return (keyword.compareTo(item.getKeyword()));
    }

    /**
     * Constructor: creates a new Item
     * @param: t is the title of Item
     * @param: key_word is the keyword of this item
     * @param: list is a list of choices for this item
     * @param: e is the content Element of this Item
```

```
 */
public Item(String t, String key_word, Element e, LinkedList list){
   title = t;
   content = e;
   keyword = key_word;
   selection = null;
   choicelist = list;
   selectedPresentation = null;
}


/**
 * Constructor: creates a new Item
 * @param: t is the title of Item
 * @param: key_word is the keyword of this item
 */
public Item(String t, String key_word){
   title = t;
   content = null;
   keyword = key_word;
   selection = null;
   choicelist = new LinkedList();
   selectedPresentation = null;
}

/**
 * Constructor: creates a new Item
 * @param: t is the title of Item
 */
public Item(String t){
   title = t;
   content = null;
   keyword = null;
   selection = null;
   choicelist = new LinkedList();
   selectedPresentation = null;
}

/**
 * Constructor: creates a new Item
 * @param: none
 */
public Item(){
   title = null;
   content = null;
   keyword = null;
   selection = null;
   choicelist = new LinkedList();
   selectedPresentation = null;
}


/**
 * returns the title of this Item
 * @param: none
 */
public String getTitle(){
   return title;
}

/**
 * returns the content element of this Item
 * @param: none
```

```
 */
public Element getContent(){
   return content;
}


/**
 * returns the keyword of this Item
 * @param: none
 */
public String getKeyword(){
   return keyword;
}


/**
 * returns the selected choice of this Item
 * @param: none
 */
public String getSelection(){
   return selection;
}

/**
 *  returns the selected presentstion element
 *  @param: none
 */
public Element getSelectedPresentation(){
   return       selectedPresentation;
}

/**
 * returns a list of choices of this Item
 * @param: none
 */
public LinkedList getChoicelist(){
   return choicelist;
}


/**
 *  returns the choice with key as its keyvalue if exists, null otherwise
 *  @param: key is the keyvalue of the choice to be find
 */
public Choice getChoice(String key){

   for(int i = 0; i<choicelist.size(); i++){
       Choice choice = (Choice)(choicelist.get(i));
       String value = choice.getKeyvalue();
       if(value.equals(key))
          return choice;
   }
   return null;
}


/**
 *  Sets the selected presentstion element to be e
 *  @param: e is the presentation element to be set
 */
public void setSelectedPresentation(Element e){
   selectedPresentation = e;
}
```

```
/**
 * Sets the title of this Item
 * @param: t is the title of this Item
 */
public void setTitle(String t){
   title = t;
}


/**
 * Sets the content of this Item
 * @param: con is the content of this Item
 */
public void setContent(Element con){
   content = con;
}


/**
 * Sets the keyword of this Item
 * @param: word is the keyword of this Item
 */
public void setKeyword(String word){
   keyword = word;
}


/**
 * Sets the Selection and the selectedPresentation of this Item
 * @param: s is the keyvalue of the selected Choice of this Item
 */
public void setSelection(String s){
   selection = s;
   selectedPresentation = getChoice(s).getPresentation();
}


/**
 *sets the default selection to be the first choice in the choicelist
 *     if choicelist is not empty
 * @param: none
 */
public void setDefaultSelection(){
   if (choicelist.size()==0)
       return;
   Choice choice =(Choice) (choicelist.get(0));
   selection = choice.getKeyvalue();
   selectedPresentation = choice.getPresentation();
}


/**
 * Sets the default default presentation to be the first choice's
 * presentation in the choicelist if choicelist is not empty
 * @param: none
 */
/*
public void setDefaultPresentation(){
   if (choicelist.size()==0)
       return;
   Choice choice =(Choice) (choicelist.get(0));
```

```
   selectedPresentation = choice.getPresentation();
}
*/

/**
 * Adds a choice to the choice list of this Item
 * @param: choice is a choice to be added to this Item
 */
public void setChoice(Choice choice){
   choicelist.addLast(choice);

}



/**
 * Returns true if subkeyword is a substring of the keyword of this Item,
 *         false otherwise
 * @param: subkeyword is a String
 */
public boolean contains(String subkeyword){
   int dif = keyword.length() - subkeyword.length() ;
   for(int j=0; j<= dif; j++){
       if (keyword.substring(j).startsWith(subkeyword.toUpperCase())){
          return true;

       }

   }
   return false;
}


/**
 *Overrides the toString method in Object
 *@return: a String of the information of the Item
 *@param: none
 */
public String toString(){
   String ret;
   ret = "title: " + title +"\n" +
         "content: " +content +"\n" +
         "keyword: " + keyword +"\n" +
         "selection: " + selection + "\n" +
         "choicelist:" +"\n";
   int size = choicelist.size();
   for (int i=0; i<size; i++){
       Choice choice = (Choice)(choicelist.get(i));
       ret = ret + "\t"+choice.toString();
   }
   ret = ret + "\n";
   return ret;
}


/**
 * Displays the information of the this Item object
 * @param: none
 */
public void print(){
   System.out.println("title: " + title);
   System.out.println("content: " +content);
   System.out.println("keyword: " + keyword);
```

```
        System.out.println("selection: " + selection);
        System.out.println("choicelist:");
        int size = choicelist.size();
        for (int i = 0; i < size; i++){
            System.out.println(choicelist.get(i).toString());
        }
    }


    /**
     * A test drive
     */
    public static void main(String[] args){
        Item item = new Item("tangent of x", "TANGENT");
        Choice c1 = new Choice("tan1.gif", "1", null);
        Choice c2 = new Choice("tan2.gif", "2", null);
        Choice c3 = new Choice("tan3.gif", "3", null);
        item.setChoice(c1);
        item.setChoice(c2);
        item.setChoice(c3);
        item.print();
    }

}
```

B.10 ItemPanel.java

```
package Project.MathTools;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

/**
 * Defines the common behaviors of RadioButtonPanel and
 * ComboBoxItemPanel classes
 */
public abstract class ItemPanel extends JPanel{
     protected JTextField textField;
     protected Item item;

    /**
     * returns the text field in this panel
     */
    public abstract JTextField getTextField();
}
```

B.11 MathNotationsFrame.java

```
package Project.MathTools;

import org.apache.xerces.parsers.DOMParser;
import org.w3c.dom.*;
import org.xml.sax.SAXException;
import org.apache.xml.serialize.*;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

import java.util.*;
import java.io.*;

/**
 * A  MathNotationsFrame object is an interface for the Notation Selection
 * Tool. It displays all the selected mathematical items with their notations.
 * The items are grouped into catalogs. It has two styles: Radio Button style
 * and Dropdown style. This interface also provides some functional Buttons
 * for the user to save their choices, find some math items, display the math
 * objects, or close the interface. It can also let the user to specify the
 * input and output files.
 */

public class MathNotationsFrame extends JFrame implements ActionListener{
    private LinkedList catalogList;
    private JPanel top;
    private JTabbedPane jtp;
    private FindFrame find_frame;
    private JTextField inStylesheetText;
    private JTextField outStylesheetText;
    private JTextField inMathmlText;
    private JTextField outMathmlText;
    private boolean isComboBox = true;
    public static boolean isSaved = false;


    /**
     * Default constructor
     */
    public MathNotationsFrame(){}


    /**
     *  Constructor: constructs a MathNotationsFrame object.
     *  @param: list is a list of Catalogs to be displayed.
     */
    public MathNotationsFrame(LinkedList list){
       super("Math Notations");
       catalogList = list;

       setLocation(200, 200);
       setSize(800, 700);

       addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e){
              //saveChoices(items);
```

```
            System.exit(0); }
      });


      createTopPanel();
      JPanel middle = createMiddlePanel();
      JPanel bottom = createBottomPanel();

      Container contents = getContentPane();
      contents.add(top, "North");
      contents.add(middle, "Center");
      contents.add(bottom, "South");

   }


   /**
    *  Creates the top panel which contains all math Items and their
    *  notations. The default style is Drop Down style.
    *  @param: none
    *  @return: none
    */
   protected void createTopPanel(){
      top = new JPanel();
      jtp = createRBTabbedPane();
      top.add(jtp);
   }


   /**
    *  Creates the middle panel of this frame. The middle panel shows
    *  the interface style options, and the text fields for input and
    *  output files.
    *  @param: none
    *  return: a JPanel object
    */
   protected JPanel createMiddlePanel(){
      JPanel middle = new JPanel();
      middle.setLayout(new GridLayout(5, 1, 1, 0));


      // create the style panel and add it to middle
      JLabel style = new JLabel("Style: ");

      ButtonGroup group = new ButtonGroup();

      JRadioButton button1 = new JRadioButton("Drop Down");
      button1.setActionCommand("ComboBox");
      button1.setSelected(false);
      button1.addActionListener(this);

      JRadioButton button2 = new JRadioButton("Radio Button");
      button2.setSelected(true);
      button2.setActionCommand("RadioButton");
      button2.addActionListener(this);

      group.add(button2);
      group.add(button1);

      JPanel stylePanel = new JPanel();
      stylePanel.add(style);
      stylePanel.add(button2);
      stylePanel.add(button1);
```

```
   middle.add(stylePanel);

   //create input stylesheet panel and add it to middle
   JPanel inStylesheetPanel = new JPanel();
   inStylesheetPanel.setPreferredSize(new Dimension(600, 60));
   JLabel inStylesheetLabel = new JLabel("Input Stylesheet file: ");
   inStylesheetText = new JTextField(30);
   inStylesheetPanel.add(inStylesheetLabel);
   inStylesheetPanel.add(inStylesheetText);
   middle.add(inStylesheetPanel);

   //create input mathml panel and add it to middle
   JPanel inMathmlPanel = new JPanel();
   inMathmlPanel.setPreferredSize(new Dimension(600, 60));
   JLabel inMathmlLabel = new JLabel("Input Mathml file: ");
   inMathmlText = new JTextField(30);
   inMathmlPanel.add(inMathmlLabel);
   inMathmlPanel.add(inMathmlText);
   middle.add(inMathmlPanel);

   //create output stylesheet panel and add it to middle
   JPanel outStylesheetPanel = new JPanel();
   outStylesheetPanel.setPreferredSize(new Dimension(600, 60));
   JLabel outStylesheetLabel = new JLabel("Output Stylesheet file: ");
   outStylesheetText = new JTextField(30);
   outStylesheetPanel.add(outStylesheetLabel);
   outStylesheetPanel.add(outStylesheetText);
   middle.add(outStylesheetPanel);

   //create output mathml panel and add it to middle
   JPanel outMathmlPanel = new JPanel();
   outMathmlPanel.setPreferredSize(new Dimension(600, 60));
   JLabel outMathmlLabel = new JLabel("Output Mathml file: ");
   outMathmlText = new JTextField(30);
   outMathmlPanel.add(outMathmlLabel);
   outMathmlPanel.add(outMathmlText);
   middle.add(outMathmlPanel);

   return middle;
}


/**
 *  Creates the bottom panel of this frame. The bottom panel shows
 *  some functional buttons: Sava, Find, Display, and Close.
 *  @param: none
 *  return: a JPanel object
 */
protected JPanel createBottomPanel(){
   JPanel bottom = new JPanel();

   JButton load = new JButton("Load");
   load.addActionListener(this);

   JButton save = new JButton("Save");
   save.addActionListener(this);

   JButton close = new JButton("Close");
   close.addActionListener(this);

   JButton find = new JButton("Find");
   find.addActionListener(this);
```

```
   JButton display = new JButton("Display");
   display.addActionListener(this);

   bottom.add(load);
   bottom.add(save);
   bottom.add(find);
   bottom.add(display);
   bottom.add(close);

   return bottom;
}

/**
 *  Gets the top panel of this frame(for the FindFrame class to use).
 *  @param: none
 *  @return: the top panel of this frame
 */
public JPanel getTop(){
   return top;
}


/**
 *  Gets the list of Catalogs(for the FindFrame class to use).
 *  @param: none
 *  @return: a LinkedList object which is a list of Catalogs
 */
public LinkedList getCatalogList(){
   return catalogList;

}


/**
 *  Gets the Tabbled pane which is contained in the top panel of this
 *  frame(for the FindFrame class to use).
 *  @param: none
 *  @return: a JTabbledPane containing all the Catalogs
 */
public JTabbedPane getTabbedPane(){
   return jtp;
}



/**
 *  Sets the find_frame to be null when the find_frme window is closed
 *  @param: none
 *  @return: none
 */
public void disposeFindFrame(){
   find_frame = null;
}


/**
 *  Creates a Tabbed pane in which the notations displayed as radio buttons
 *  @param: none
 *  @return: a JTabbedPane object in which the notations displayed as
 *  radio buttons
 */
public JTabbedPane createRBTabbedPane(){
```

```
   isComboBox = false;
   JTabbedPane tp = new JTabbedPane();

   Catalog cat;
   LinkedList itemList;
   JScrollPane  pane;
   String name;

   for(int i = 0; i<catalogList.size(); i++){
       String tip = "";
       cat = (Catalog)(catalogList.get(i));
       name = cat.getName();
       itemList = cat.getItemList();
       for(int n=0; n<itemList.size(); n++){
          Item it = (Item)(itemList.get(n));
          tip = tip + it.getKeyword() +"   ";
       }

       //    CatalogPaneFactory  factory = new CatalogPaneFactory(cat);
       //pane = factory.getScrollPane();
        JPanel panel = new RadioButtonCatalogPanel(cat);

        pane = new JScrollPane
          (panel,
           ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS,
           ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
       pane.setPreferredSize(new Dimension(700, 300));
       tp.addTab(name, null, pane, tip);

   }

   return tp;

}



/**
 *  Creates a Tabbed pane in which the notations displayed as drop down box
 *  @param: none
 *  @return: a JTabbedPane object in which the notations displayed in
 *  drop down boxes
 */
public JTabbedPane createCBTabbedPane(){
   isComboBox = true;
   JTabbedPane tp = new JTabbedPane();

   Catalog cat;
   LinkedList itemList;
   JScrollPane  pane;
   String name;

   for(int i = 0; i<catalogList.size(); i++){
       String tip = "";
       cat = (Catalog)(catalogList.get(i));
       name = cat.getName();
       itemList = cat.getItemList();
       for(int n=0; n<itemList.size(); n++){
          Item it = (Item)(itemList.get(n));
          tip = tip + it.getKeyword() +"   ";
       }

       //    CatalogPaneFactory  factory = new CatalogPaneFactory(cat);
```

```
        //pane = factory.getScrollPane();
         JPanel panel = new ComboBoxCatalogPanel(cat);

         pane = new JScrollPane
           (panel,
            ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS,
            ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
        pane.setPreferredSize(new Dimension(700, 300));
        tp.addTab(name, null, pane, tip);

    }

    return tp;

}

/**
 * Loads the saved choices from file Choice.dat
 * @param: none
 * @return: none
 */
public void loadChoices(){
    try{
        FileReader inFile = new FileReader("Choice.dat");
        BufferedReader inBuffer = new  BufferedReader( inFile );
        int n = Integer.parseInt(inBuffer.readLine());
        for( int i = 0; i<n; i++){
           String s = inBuffer.readLine();
           StringTokenizer t = new StringTokenizer(s, "=");
           String keyword = t.nextToken();
           String keyvalue = t.nextToken();

           for(int j=0; j<catalogList.size(); j++){
               Catalog cat = (Catalog)(catalogList.get(j));
               LinkedList itemList = cat.getItemList();
               for(int k=0; k<itemList.size(); k++){
                  Item item = (Item)(itemList.get(k));
                  if(keyword.equals(item.getKeyword()))
                      item.setSelection(keyvalue);

               }
           }
        }
    }

    catch(IOException e){
         JOptionPane.showMessageDialog
             (null, "Nothing to be loaded!",
              "Warning", JOptionPane.WARNING_MESSAGE);

        //System.out.println("Nothing to be loaded!");
    }
}

/**
 *  Implements the interface ActionListener
 */
public void actionPerformed(ActionEvent evt){
    String command = evt.getActionCommand();

    if (command.equals("Load")){ //click on the Load button
        loadChoices();
        if (isComboBox){
```

```
            top.removeAll();
            jtp = createCBTabbedPane();
            top.add(jtp);
            validate();
        }
        else{
            top.removeAll();
            jtp = createRBTabbedPane();
            top.add(jtp);
            validate();

        }
    }

    else if (command.equals("Save")){ //click on the save button
        Object[] options = {"Save keyvalue",
                            "Save stylesheet", "Save both"};
        int n = JOptionPane.showOptionDialog
           (this, "Save keyvalue, stylesheet, or both?",
            "Save", JOptionPane.YES_NO_CANCEL_OPTION,
            JOptionPane.QUESTION_MESSAGE,null, options, options[2]);

        if (n == JOptionPane.YES_OPTION) {
           saveChoices();
        }
        else if (n == JOptionPane.NO_OPTION) {
           saveStyleSheet();
        }
        else if (n == JOptionPane.CANCEL_OPTION) {
           saveChoices();
           saveStyleSheet();
        }
        else;

    }

    else if (command.equals("Display")){//click on the Display button

        String inFile;
        String outFile;
        String newStyleFile;
        String text;

        saveStyleSheet();

        //get the output stylesheet file name.
        //The default output file name is mml2mml.new
        text = outStylesheetText.getText();
        if (text != null && !text.equals("") )
           newStyleFile = text;
        else newStyleFile = "mml2mml.new";


        //get the input content mathML file name.
        //The default file name is show.xml
        text = inMathmlText.getText();
        if (text != null && !text.equals("") )
           inFile = text;
        else inFile = "show.xml";


        //get the output Presentation MathML file name.
        //The default output file name is show.out
```

```
            text = outMathmlText.getText();
            if (text != null && !text.equals("") )
               outFile = text;
            else outFile = "show.out";

            Runtime run = Runtime.getRuntime();

            String xt_cmd = "xt" + " " + inFile + " " + newStyleFile + "
"+outFile;
            String cd_cmd1 = "cd" + " " + "../../gtkmathview-dliu/viewer";
            String view_cmd = "viewer " + "/scl/people/dliu3/Project/MathTools/"+
outFile;
            String cd_cmd2 = "cd "+ "../../Project/MathTools";

            try{
                  File f = new File(".display.sh");
                  FileOutputStream fs = new FileOutputStream(f);
                  PrintStream target = new PrintStream(fs);
               target.println(xt_cmd);
               target.println(cd_cmd1);
               target.println(view_cmd);
               target.println(cd_cmd2);

               String[] cmd = {"/bin/sh", ".display.sh"};
               run.exec(cmd);
               //run.exec("cd");
               //run.exec(cd_cmd1, null);
               //run.exec(view_cmd);
               //run.exec("cd");
               //run.exec( cd_cmd2, null);
            }

            catch(IOException e){
               System.out.println(e);
            }


        }

        else if (command.equals("Close")){//click on the Close button
            if (MathNotationsFrame.isSaved == false){
               int n = JOptionPane.showConfirmDialog(
                           null, "Save the changes?",
                               "An confirm Question",
                               JOptionPane.YES_NO_OPTION);
                     if (n == JOptionPane.YES_OPTION) {
                     saveChoices();
                    }
            }
            System.exit(0);
        }

        else if (command.equals("Find")){ //click on the Find button
            if(find_frame == null)
               find_frame = new FindFrame(this);
            else find_frame.setVisible(true);
            //return;

        }

        else if (command.equals("ComboBox")){//choose the drop down box style
            top.removeAll();
            jtp = createCBTabbedPane();
```

```
       top.add(jtp);
       validate();
   }

   else if (command.equals("RadioButton")){//choose radio button style
       top.removeAll();
       jtp = createRBTabbedPane();
       top.add(jtp);
       validate();
   }
   else;
}



/**
 *  Saves the key values of selected notations
 *  @param: none
 *  @return: none
 */
public void saveChoices(){
    try
        {
            File f = new File("Choice.dat");
            FileOutputStream fs = new FileOutputStream(f);
            PrintStream target = new PrintStream(fs);

         Catalog cat;
         String name;
         LinkedList itemList;
         Item item;


         //count how many items
         int count = 0;
         for (int i=0; i<catalogList.size(); i++){
             cat = (Catalog)(catalogList.get(i));
             itemList = cat.getItemList();
             count = count + itemList.size();
         }


         // save choices
         target.println(count);
            for (int i=0; i<catalogList.size(); i++){
             cat = (Catalog)(catalogList.get(i));
             //  name = cat.getName();
             //target.println("Catalog: " + name);
             itemList = cat.getItemList();
             for(int j=0; j<itemList.size(); j++){
                item = (Item)(itemList.get(j));
                String keyword = item.getKeyword();
                String selection = item.getSelection();
                target.println(keyword +"=" + selection);
             }
         }
            target.close();
         MathNotationsFrame.isSaved = true;
        }

    catch(IOException e)
        {
            System.out.println(e);
```

```
        }
}



/**
 *  Saves the stylesheet for the selected notations
 *  @param: none
 *  @return: none
 */
public void saveStyleSheet(){
    try
        {
            File f = new File("mml.out");
            FileOutputStream fs = new FileOutputStream(f);
            PrintStream target = new PrintStream(fs);

         OutputFormat format = new OutputFormat();
         XMLSerializer out = new XMLSerializer(fs, format);

         //target.println();

         Catalog cat;
         String name;
         LinkedList itemList;
         Item item;

            for (int i=0; i<catalogList.size(); i++){
             cat = (Catalog)(catalogList.get(i));
             name = cat.getName();
             target.println("\n\n\n <!-- Catalog: " + name + "-->\n\n");
             itemList = cat.getItemList();
             for(int j=0; j<itemList.size(); j++){
                item = (Item)(itemList.get(j));
                String keyword = item.getKeyword();
                String selection = item.getSelection();
                Element present = item.getSelectedPresentation();
                Element content = item.getContent();
                //System.out.println(selection);
                //target.println("\n"+keyword + " = " + selection);
                NodeList nodelist =
                    present.getElementsByTagName("xsl:template");
                int length = nodelist.getLength();

                try {
                    //out.serialize(content);
                    target.println("\n");
                    for(int k=0; k<length; k++){
                       Node anode = nodelist.item(k);
                       out.serialize((Element)anode);

                    }
                }
                catch (IOException ioe) {
                    ioe.printStackTrace();
                }
             }
         }
            target.close();
         MathNotationsFrame.isSaved = true;
         createNewStylesheet();
        }
```

```
        catch(IOException e)
            {
                System.out.println(e);
            }
    }



    public void createNewStylesheet(){
       String inStyleFile;
       String outStyleFile;
       String text;

       //get the input stylesheet file name.
       //The default file name is mml2mml.xsl
       text = inStylesheetText.getText();
       if (text != null && !text.equals("") )
           inStyleFile = text;
       else inStyleFile = "mml2mml.xsl";


       //get the output stylesheet file name.
       //The default output file name is mml2mml.new
       text = outStylesheetText.getText();
       if (text != null && !text.equals("") )
           outStyleFile = text;
       else outStyleFile = "mml2mml.new";

        try{
            FileReader inFile1 = new FileReader(inStyleFile);
            BufferedReader inBuffer1 = new  BufferedReader( inFile1 );

            FileReader inFile2 = new FileReader("mml.out");
            BufferedReader inBuffer2 = new  BufferedReader( inFile2 );

          FileWriter outFile = new FileWriter(outStyleFile);
          BufferedWriter outBuffer = new BufferedWriter(outFile);

          String line;
          while((line = inBuffer1.readLine())!= null){
             if(!(line.trim()).equals("</xsl:stylesheet>"))
                 outBuffer.write(line);
             outBuffer.write("\n");
          }

          inBuffer1.close();

          String extra = "<?xml version="+'"'+"1.0"+'"' +" encoding="+'"'+"UTF-
8" +'"'+"?>";
          while((line = inBuffer2.readLine())!= null){
             if(!(line.trim()).equals(extra))
                 outBuffer.write(line);
             outBuffer.write("\n");
          }

          outBuffer.write("</xsl:stylesheet>");
          inBuffer2.close();

          outBuffer.close();

        }
```

```
        catch(IOException e){
                System.out.println(e);
        }

    }


    /**
     *  for debugging
     *  @param: none
     *  @return: none
     */
    void printClassName(Object obj) {
        System.out.println("The class of " + obj +
                           " is " + obj.getClass().getName());
    }
}
```

B.12 NotationSelection.java

```
package Project.MathTools;


import org.apache.xerces.parsers.DOMParser;
import org.w3c.dom.*;
import org.xml.sax.SAXException;
import java.io.IOException;
import org.apache.xml.serialize.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
import java.io.*;

/**
 * NotationSelection class is the engine class which contains the main
 * method for the Notation Selection Too.
 */
public class NotationSelection
{
    public static void main(String[] args)
    {
       String xmlFile = "../mnotations.mss";
       DOMParser parser = new DOMParser();

       try {
           parser.parse(xmlFile);
       }
       catch (SAXException se) {
           se.printStackTrace();
       }
       catch (IOException ioe) {
           ioe.printStackTrace();
       }

       Document doc = parser.getDocument();

       DataReader reader = new DataReader(doc);
       SortedList list = reader.getCatalogs();

       //set default selection to be the first choice if any.
       for(int i=0; i<list.size(); i++){
           Catalog cat = (Catalog)(list.get(i));
           LinkedList itemList = cat.getItemList();
           for(int j=0; j<itemList.size(); j++){
              Item item = (Item)(itemList.get(j));
              item.setDefaultSelection();
              //item.setDefaultPresentation();
           }
       }


       JFrame frame = new MathNotationsFrame(list);
       frame.pack();
       frame.setVisible(true);


    }
}
```

B.13 RadioButtonCatalogPanel.java

```
package Project.MathTools;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

import java.util.*;

/**
 *  A RadioButtonCatalogPanel object contains a set of RadioButtonItemPanel
 *  objects and shows all the items and their choices of a catalog. All
 *  choices are orgnized in a group of radio buttons
 */
public class RadioButtonCatalogPanel extends CatalogPanel{
    /**
     *  Constructor: initializes a panel for a catalog. The panel uses GridBag
     *               layout.
     *  @param: cat is a Catalog object corresponding this panel
     */
    public RadioButtonCatalogPanel(Catalog cat){
       catalog = cat;
       addItemPanels();

    }

    /*
     * Creates and adds all ItemPanels to this Catalog panel.
     * @return: none
     * @parm: none
     */
    protected void addItemPanels(){
       LinkedList list = catalog.getItemList();
       int size = list.size();

        GridBagLayout gridbag = new GridBagLayout();
        GridBagConstraints c = new GridBagConstraints();
        setLayout(gridbag);
       //setAlignmentY(10);
       c.gridwidth = 1;
       c.anchor = GridBagConstraints.WEST;
       c.insets = new Insets(0, 10, 0, 10);

       Item item;
       for(int i=0; i<list.size(); i++){
           item = (Item)(list.get(i));
           JPanel panel = new RadioButtonItemPanel(item);
           panel.setMaximumSize( new Dimension(600, 60));
           panel.setPreferredSize(new Dimension(600, 60));
           //panel.setBackground(Color.lightGray);
            c.weightx = 0.9;
            c.gridx = 0;
            c.gridy = i;
            gridbag.setConstraints(panel, c);
           add(panel);
       }
    }

}
```

B.14 RadioButtonItemPanel.java

```
package Project.MathTools;

import org.w3c.dom.*;
import org.apache.xerces.parsers.DOMParser;


import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

import java.util.*;

/**
 * A RadioButtonItemPanel contains a text field and a group of radio buttons
 * which contains a list of choices. It corresponds a math item
 */
public class RadioButtonItemPanel extends ItemPanel
    implements ActionListener{
    //private Item item;
    private ButtonGroup group;
    private GridBagLayout gridbag;
    private GridBagConstraints c;
    // private JTextField textField;

    /**
     * default constructor
     */
    public RadioButtonItemPanel(){}

    /**
     *  Constructor: create a RadioButtonItemPanel object
     *  @param: it is a math item
     */
    public RadioButtonItemPanel(Item it){

       item = it;

       gridbag = new GridBagLayout();
       c = new GridBagConstraints();
        setLayout( gridbag);
       //setMaximumSize(new Dimension(700, 60));
       //setPreferredSize(new Dimension(600, 60));
       addTextField();
       textField.setBorder(null);
       //addChoiceBox();
       addChoiceButtons();
    }


    /**
     *  Creates the text field and adds it to this panel
     *  @param: none
     *  @return: none
     */
    protected void addTextField(){
       String name = item.getKeyword();
       textField = new JTextField(name + ":  ");
       textField.setEditable(false);
```

```
   c.gridx = 0;
   c.gridy = 0;
   c.gridwidth = 2;
   c.gridheight = 1;
    c.anchor = GridBagConstraints.WEST;
   gridbag.setConstraints(textField, c);

   add(textField);
}



/**
 *  returns the text field
 *  @param: none
 *  @return: the text field
 */
public JTextField getTextField(){
   return textField;

}


/**
 *  Creates the group of choice buttons and adds it to this panel
 *  @param: none
 *  @return: none
 */
protected void addChoiceButtons(){
   group = new ButtonGroup();
   LinkedList choicelist = item.getChoicelist();

   //find  the selected choice index in the choice list of this item
   String key = item.getSelection();


   c.weightx = 0.5;
    for(int i = 0; i < choicelist.size(); i++){
        Choice choice = (Choice)(choicelist.get(i));
        String imagefile = choice.getImage();
        String  keyvalue = choice.getKeyvalue();
        boolean isSelected = false;
        if (keyvalue.equals(key)){
            isSelected = true;
         // item.setSelection(keyvalue);
        }
        ImageIcon icon = new ImageIcon("../images/" + imagefile);
        while(icon.getImageLoadStatus() != MediaTracker.COMPLETE)
            Thread.yield();
        JRadioButton button = new JRadioButton();
        button.setSelected(isSelected);
        button.addActionListener(this);

        group.add(button);
        JLabel image = new JLabel(icon);

        //c.weightx = 0;
        c.gridx = 2 + i*4;
        c.gridy = 0;
        c.gridwidth = 1;
        c.anchor = GridBagConstraints.EAST;
        gridbag.setConstraints(button, c);
```

```
            add(button);
            button.setActionCommand(keyvalue);

           //c.weightx = 0;
            c.gridx = 3 + i*4;
            c.gridy = 0;
            c.gridwidth = 3;
            c.anchor = GridBagConstraints.WEST;
            gridbag.setConstraints(image, c);
            add(image);

        }


    }



    /**
     *  Implements the interface ActionListener
     */
    public void actionPerformed(ActionEvent evt){
        String command =evt.getActionCommand();
        item.setSelection(command);
       //LinkedList choicelist = item.getChoicelist();
       //Choice choice = item.getChoice(command);
       //Element e = choice.getPresentation();
       //item.setSelectedPresentation(e);

       MathNotationsFrame.isSaved = false;
    }

}
```

B.15 SortedList.java

```
package Project.MathTools;

import java.util.LinkedList;

/**
 *  A SortedList is a list of Comparable objects in ascending order
 */
public class SortedList extends LinkedList{

    /**
     * Constructor: initializes a SortedList object
     */
    public SortedList(){
       super();
    }


    /**
     *  Insert o in the right place
     *  @param: o is a Comparable object to be inserted
     *  @return: none
     */
    public void insert(Comparable o){
        if (size()==0)
            addFirst(o);
        else if (o.compareTo(getFirst()) < 0)
            addFirst(o);
        else
            insertAfter(0, o);
    }

    /**
     *  Begins from index here to recursively find the right position and
     *  insert o.
     *  @param: here is the index after which o is inserted somewhere
     *  @param: o is the object to be inserted
     *  @return: none
     */
    protected void insertAfter(int here, Comparable o){
        int next = here+1;
        if(next == size())
            addLast(o);
        else if (o.compareTo(get(next)) < 0)
            add(next, o);
        else
            insertAfter(next, o);
    }


    /**
     *  Displays all the object in the list
     *  @param: none
     *  @return: none
     */
    public void displayAll(){
        if (size()<=0){
            System.out.println("The list is empty");
            return;
        }
```

```
        else {
           for(int i=0; i<size(); i++){
              System.out.println(get(i).toString());
           }

        }
    }



    /**
     *  A test drive
     */
    public static void main(String[] args){
       SortedList list = new SortedList();
       Item item1 = new Item("tangent of x", "TANGENT");
       Item item2 = new Item("x divides y", "DIVISION");
       Item item3 = new Item("x times y", "MULT");
       list.insert(item1);
       list.insert(item2);
       list.insert(item3);

       list.displayAll();

    }

}
```